

# HIBAJELZÉS, HIBAJAVÍTÁS

FORRÁS: [HTTPS://GYIRES. INF. UNIDEB. HU/GYBITT/30/CH03S02. HTML](https://GYIRES. INF. UNIDEB. HU/GYBITT/30/CH03S02. HTML)

ILLETVE

PROF. DR. GALÁNTAI AURÉL JEGYZETE

# MIÉRT?

- A csatornán
  - Optikai kábel
  - Vezeték nélküli ...
- történő továbbításkor megváltozhat az üzenet.
- Ezért kibővítjük az eredeti üzenetet, hogy vételkor ellenőrizni tudjuk, történt-e elváltozás:
- Kód+redundáns információ

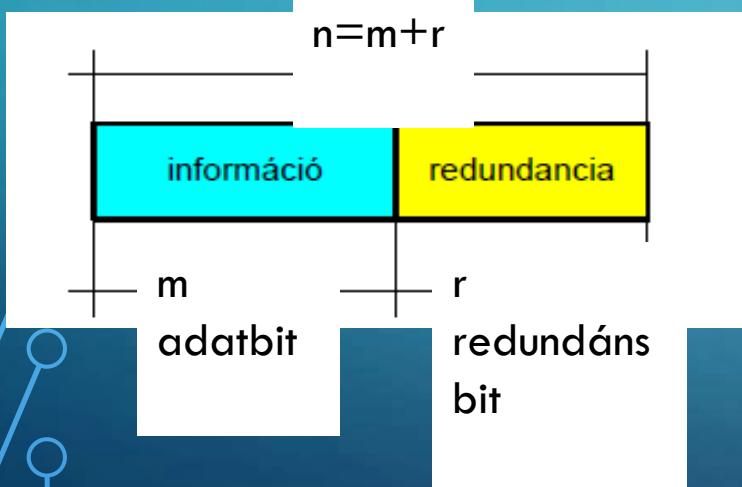
# Hogyan?

- Hibajelző (újra-küldés)
- Paritás bitek hozzáadása
- Hibajavító kód megalkotása:
  - Hamming-kódok,
  - bináris konvolúciós kódok,
- ...

# BLOKK KÓD

$x_1$	$x_2$	$x_3$	$x_4$
00	10	01	11

$x_1$	$x_2$	$x_3$	$x_4$
00000	10110	01101	11011



- A keretek
- $m$  adatbitből, vagyis üzenetbitből (message bit) és
- $r$  redundáns bitből, vagyis ellenőrző bitből (check bit) állnak.
- A **blokk-kódokban (block code)** az  $r$  ellenőrző bitek kiszámítása csak és kizárolag a hozzájuk tartozó  $m$  adatbitek függvényeként történik, pontosan úgy, mintha egy nagy táblából kikeresnénk az  $m$  adatbithez tartozó  $r$  ellenőrző bitet.
- Ha az  $m$  adatbitet közvetlenül, változtatás (előzetes kódolás) nélkül küldjük el az ellenőrző bitekkel együtt, akkor **szisztematikus kódokról (systematic code)** beszélünk.
- **Lineáris kódokban** az  $r$  ellenőrző bit az  $m$  adatbit lineáris függvénye, melyre gyakori példa a kizárt vagy (xor) vagy a modulo 2 összeadás. (Ez azt jelenti, hogy a kódolás folyamata mátrixszorzással vagy egyszerű logikai áramkörökkel végezhető.)
- lineáris, szisztematikus blokk-kódok,  $(n,m)$  kód.
- kódsebesség (code rate) vagy egyszerűen sebesség a kódszó azon része, amely a nem redundáns információt tartalmazza (tehát  $m/n$ ). Zajos csatornára akár  $1/2$  is lehet, így a vett információ fele redundancia, míg egy jó minőségű csatornára megközelítheti az 1-et, hiszen kevés ellenőrző bitet csatolnak egy hosszú üzenethez.

# MI A „HIBA”?

- 10001001
- 10110001
- 00111000

$$d_H=3$$

- Az olyan helyek számát, amelyeken a két kódszóban különböző bitek állnak, a két kódszó Hamming-távolságának [Hamming, 1950] nevezük.
- A jelentősége abban áll, hogy ha két kódszó Hamming-távolsága  $d$ , akkor  $d$  darab egy bitet érintő hiba/átalakítás kell ahhoz, hogy az egyik kódszó a másikba menjen át.
- Megjegyzés: természetesen nem csak bináris kódokra vonatkoztathatók a kimondottak.

# HAMMING TÁVOLSÁG

- Megadva az ellenőrző bitek kiszámításának módját, meg lehet alkotni a legális kódszavak teljes listáját, és ebből a listából ki lehet keresni azt a két kódszót, melyeknek legkisebb a Hamming-távolsága. Ez a távolság a teljes kód Hamming-távolsága.

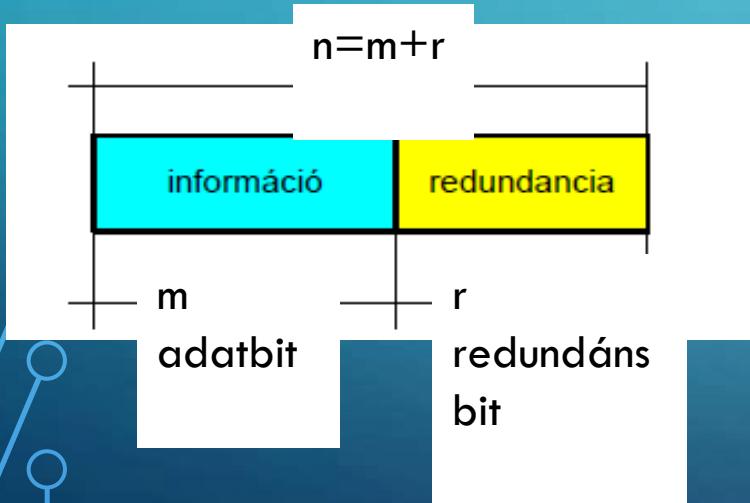
$$d = \min_{\substack{i,j \\ i \neq j}} d_H (\mathbf{u}_i, \mathbf{u}_j) .$$

# HOGYAN ÉSZLELHETÜNK HIBÁT?

- Ha  $m$  hosszúságú a bináris üzenet (információ), akkor  $2^m$  lehetséges adatüzenet legális (ennyit tudunk megalkotni), de az ellenőrző bitek kiszámítási módja miatt nem fordul elő mind a  $2^n$  lehetséges kódszó (mert a redundáns rész az  $m$  hosszúságú üzenet függvénye).

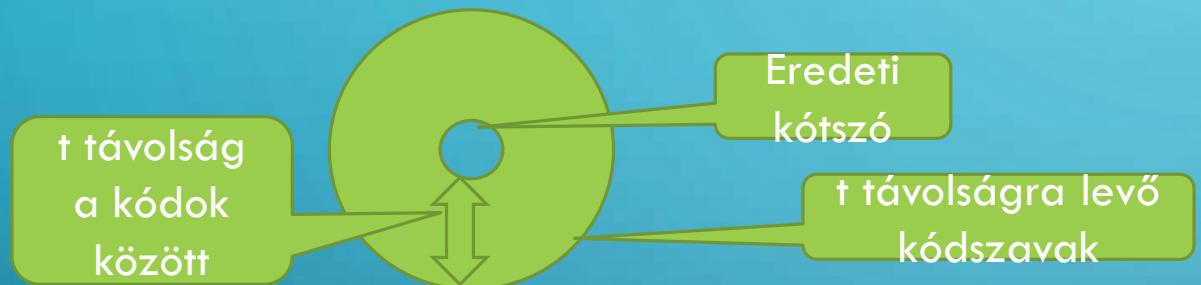
$r$  ellenőrző bit esetén csupán a lehetséges üzenetek töredéke,  $2^m / 2^n$ -ed vagy  $1 / 2^n$ -ed része lesz legális kódszó.

Ez a különbség az, amellyel az üzenet beágyazódik a kódszavak terébe, amely lehetővé teszi, hogy a vevő kijelezze és kijavítsa az átvitel során keletkezett hibákat (azaz a redundáns résszel kibővített üzenetekből több van, és könnyebb felismerni, ha nem a korrekt (érvénytelen) kódszó jön át).

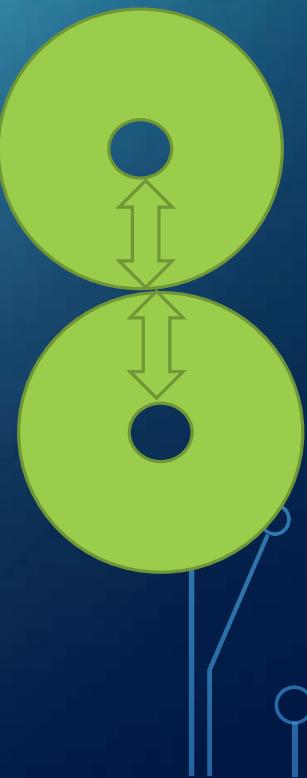


# HIBAJELZŐ? HIBAJAVÍTÓ?

- Amikor a vevő egy érvénytelen kódszót lát, tudja, hogy átviteli hiba történt.
- Az, hogy egy blokk-kód hibajelző vagy hibajavító tulajdonságú-e, a **kód Hamming-távolságától** függ.
- Ahhoz, hogy  $t$  hibát jelezni tudjunk,  $t+1$  Hamming-távolságú kód kell, mert egy ilyen kódban  $t$  bithiba nem tudja a kódszót egy másik érvényes kódszóba vinni.



- Hasonlóan: ahhoz, hogy  $t$  hibát ki tudunk javítani,  $2t+1$  Hamming-távolságú kód kell, mert így az érvényes kódszavak olyan távol vannak egymástól, hogy még  $t$  bit megváltozásakor is közelebb lesz az eredeti kódszó a hibáshoz, mint bármely másik érvényes kódszóhoz, így az egyértelműen meghatározható, (feltételezve, hogy nagyobb számú bithibáknak kicsi a valószínűsége.)
- Tehát legalább (azaz min)  $2t+1$  távolság kell legyen a két kód között, hogy biztosan el lehessen „különíteni” őket.



## PÉLDA

0000000000

0000011111

1111100000

1111111111

- $d_H=5, d=5$
- $5=2*t+1=2*2+1$
- $t=2$  bitnyi hibajavítás lehetséges.

- ha a 0000000111 kódszó érkezik, nem legális
- a vevő tudja, hogy az eredetinek 0000011111-nek kellett lennie (ez van legközelebb az érvényes kódszóhoz). ( $t=2$ )

Azonban, ha hárombitnyi hiba a 0000000000-t 0000000111-re változtatta, akkor a kódszó nem megfelelően lesz kijavítva ( $t=3$  lenne).

# ELMÉLETI MEGKÖZELÍTÉS:

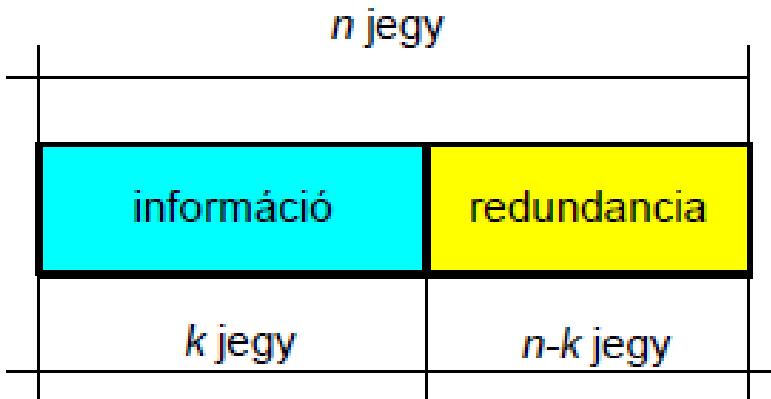
A következő jelöléseket használjuk:

$U = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M\}$  - csatorna kód,

$\mathbf{u}_i = [u_{i1}, u_{i2}, \dots, u_{in}]$  -  $i$ -edik vektor (kódszó) ( $u_{ij} \in \{0, 1\}$ )

$u_{ij}$  - az  $i$ -edik kódszó  $j$ -edik eleme,

$k = \log M$  - egy kódszó információ bitjeinek száma.



Az ilyen típusú kódokra esetenként  $(n, k)$ -kódként hivatkozunk.

**Definíció:** Két kódszó Hamming-távolságán az egymástól páronként különböző jegyek számát érjük, azaz

$$d_H (\mathbf{u}_i, \mathbf{u}_j) = |\{k \mid u_{ik} \neq u_{jk}, 1 \leq k \leq n\}|.$$

**Definíció:** Egy  $\mathbf{u}_i$  kódszó Hamming-súlyán a nemzérus komponensek számát értjük, azaz

$$W (\mathbf{u}_i) = |\{k \mid u_{ik} \neq 0, 1 \leq k \leq n\}|.$$

**Definíció:** A különböző kódszavak közti  $d$  minimális Hamming-távolságot minimális távolságnak (kód távolságnak) nevezzük, azaz

$$d = \min_{\substack{i,j \\ i \neq j}} d_H (\mathbf{u}_i, \mathbf{u}_j). \quad (3.2)$$

A Hamming-távolságra teljesül, hogy

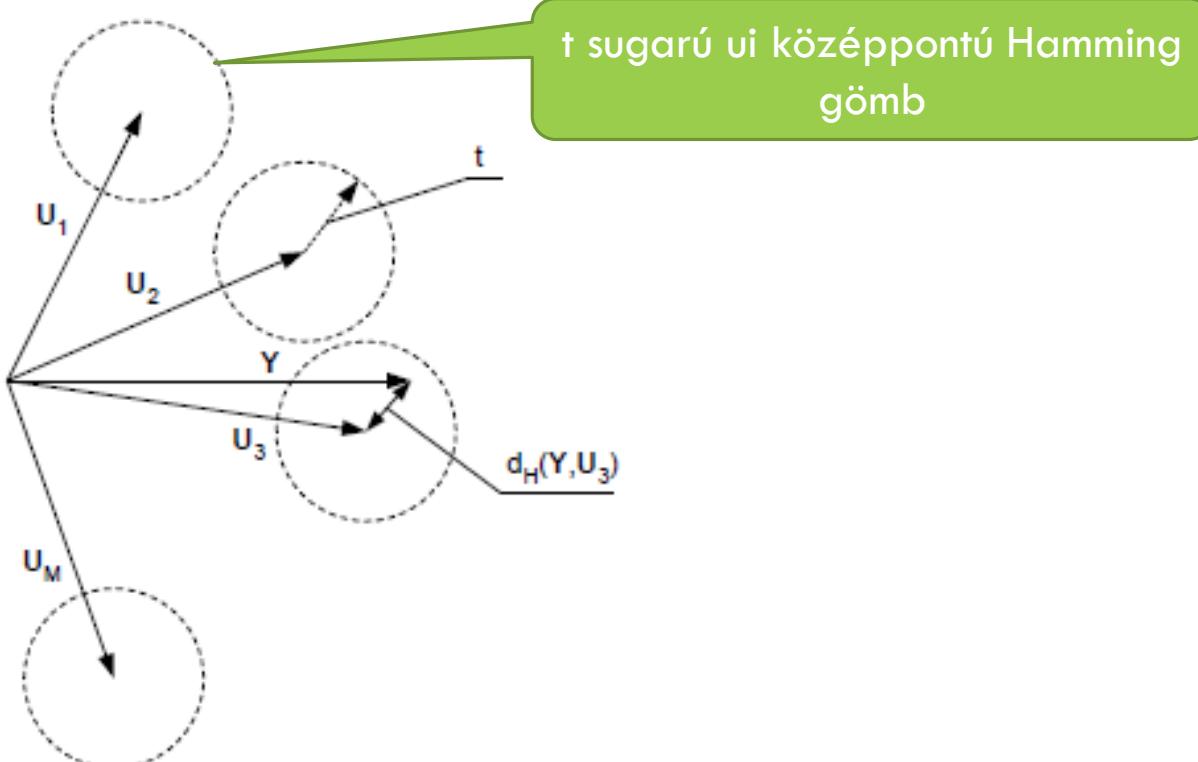
- (i)  $d_H(\mathbf{u}_i, \mathbf{u}_j) \geq 0$ ,  $d_H(\mathbf{u}_i, \mathbf{u}_j) = 0 \Leftrightarrow \mathbf{u}_i = \mathbf{u}_j$ ,
- (ii)  $d_H(\mathbf{u}_i, \mathbf{u}_j) = d_H(\mathbf{u}_j, \mathbf{u}_i)$ ,
- (iii)  $d_H(\mathbf{u}_i, \mathbf{u}_j) \leq d_H(\mathbf{u}_i, \mathbf{u}_k) + d_H(\mathbf{u}_k, \mathbf{u}_j)$ .

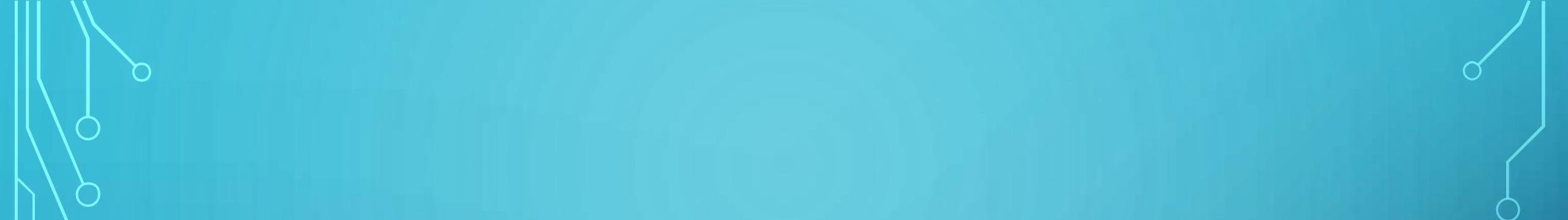
A Hamming távolság ezek szerint megfelel a vektortereken definiált normák feltételeinek, ami még fontos lesz, hiszen a bináris kódok tulajdonképpen vektorok.

**Definíció:** Egy kód  $t$  hibáig korrekt, ha bármelyik kódszóban  $t$  vagy ennél kevesebb hibás szimbólumot helyesen dekódol a minimális távolság elv alapján.

**Tétel:** Egy kód akkor és csak akkor korrekt  $t$  hibáig, ha  $d \geq 2t + 1$ .

**Bizonyítás:** A  $t$  sugarú  $u_i$  középpontú Hamming-gömböt azok a  $0-1$  elemű vektorok alkotják, amelyek Hamming-távolsága  $u_i$ -től kisebb, vagy egyenlő mint  $t$ .





**Következmény:** Egy adott kód hibajavító képessége  $t$ , és a kód minimális távolsága összefüggnek. Nevezetesen

$$t = \lfloor (d - 1) / 2 \rfloor. \quad (3.3)$$

Emlékeztető:  $[2,13] = 2$

# TOVÁBBGONDOLVA

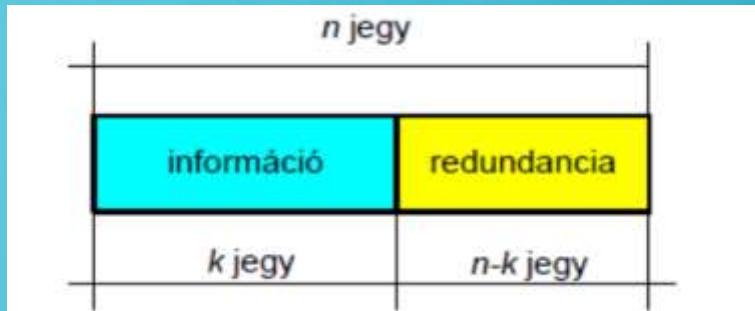
A blokk kódok három fő paramétere:

$M$  - a kódvektorok száma (kód mérete),

$n$  - a kódszavak hossza,

$d$  - a kódtávolság.

Az  $M$  paraméter helyett szokás az információs bitek  $k$  számát, vagy az  $R = k/n$  hánnyadost is használni. Szokás továbbá a blokk kódra az  $(n, M, d)$  formában is hivatkozni.



A blokk kódok három fő paramétere:

$M$  - a kódvektorok száma (kód mérete),

$n$  - a kódszavak hossza,

$d$  - a kódtávolság.

Az  $M$  paraméter helyett szokás az információs bitek  $k$  számát, vagy az  $R = k/n$  hányadost is használni. Szokás továbbá a blokk kódra az  $(n, M, d)$  formában is hivatkozni.

A jó hibajavító tulajdonság elérése adott  $M$  ( $k$  vagy  $R$ ) és  $n$  esetén a  $d$  "maximálását" jelenti. Duális formában ugyanez az  $R$  hányados ( $k$  vagy  $M$ ) maximalizálását jelenti adott  $d$  minimum távolság és  $n$  hosszúság esetén.

Lásd a 12. dián

Egy adott kódszó  $t$  sugarú Hamming-gömbjében  $\sum_{i=0}^t \binom{n}{i}$  darab vektor van. Ebből következik az alábbi

**Tétel (Hamming korlát):** Tetszőleges, legfeljebb  $t$  hibát javító blokk kódra fenn kell állnia a

$$M \sum_{i=0}^t \binom{n}{i} \leq 2^n \quad (3.4)$$

egyenlőtlenségnek.

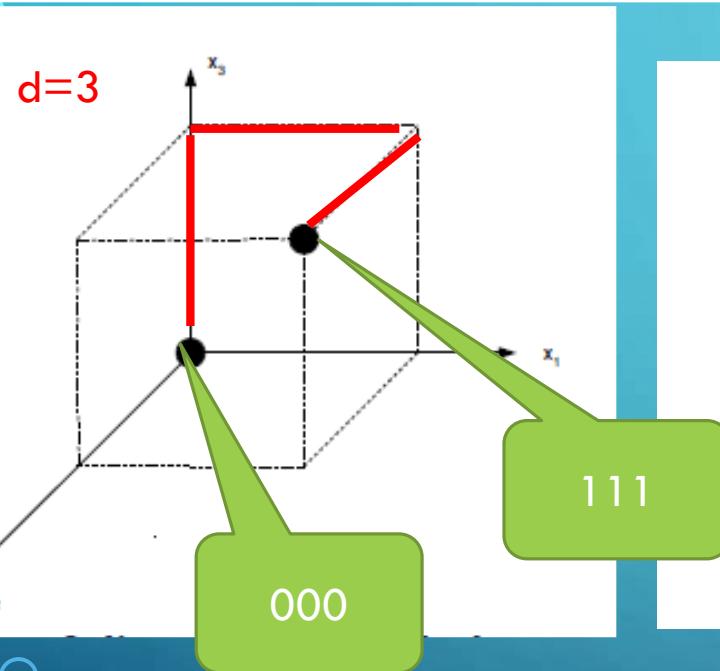
Az egyenlőséget kielégítő kódokat *perfekt kódoknak* nevezzük.

A perfekt kódok kijavítanak  $t$  vagy kevesebb hibát, de egyikük sem ismer fel, vagy javít  $t$ -nél több hibát. Geometriailag ezek a kódok a  $2^n$  darab bináris vektor  $M$  darab  $t$  sugarú gömbbel történő közös részek és hézagok nélküli, un. "*legszerosabb kitöltését*" valósítják meg. A "tökéletességük" az  $R$  maximálását jelenti adott  $d = 2t + 1$  és  $n$  esetén.

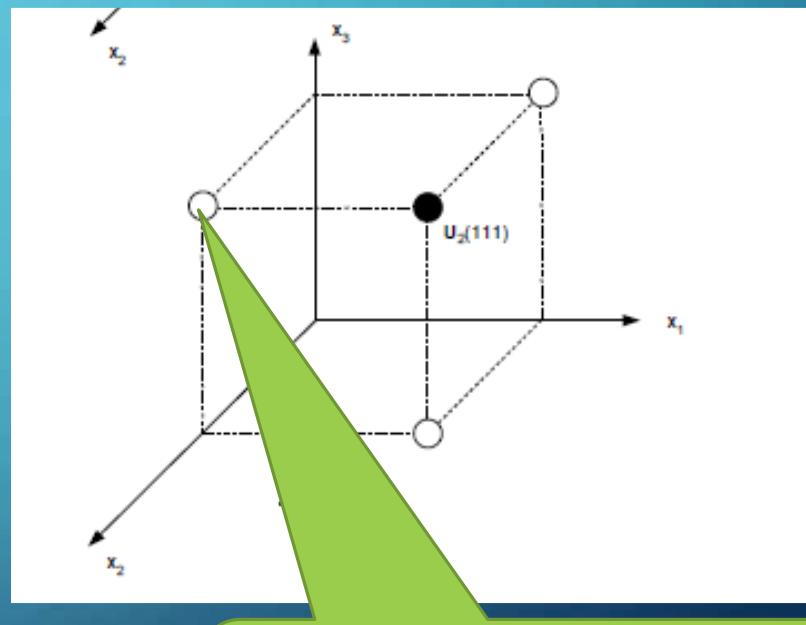
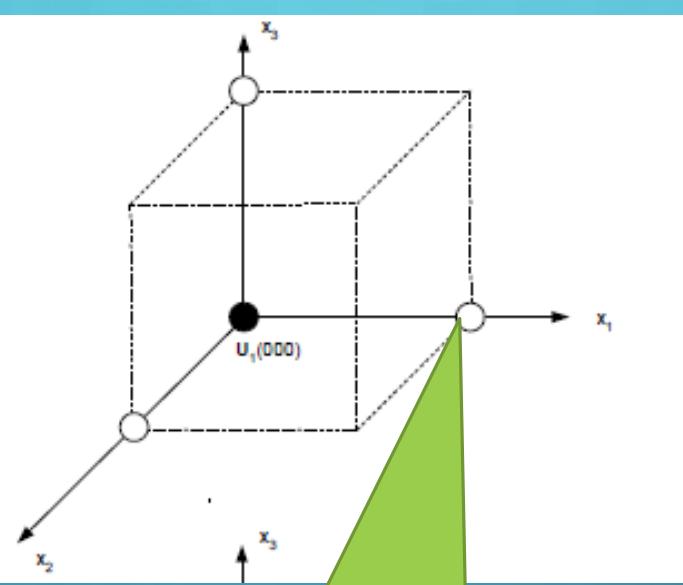
Csak három tökéletes bináris kódosztály létezik:

- páratlan hosszúságú bináris ismétlő kódok,
- Hamming kódok (1 hibát javítanak),
- Golay kód ( $n = 23$ ,  $d = 7$ , 3 hibát javít).

**Példa:** A legegyszerűbb bináris ismétlő kód hossza 3. A kód a jelet háromszor megismétli: a '0'-hoz a '000', az '1'-hez az '111' kódszót rendeli. A két kódszó távolsága  $d_H((000), (111)) = 3$ . Ugyancsak igaz, hogy  $d_{\min} = 3$ . A 3 hosszúságú bináris vektorok száma  $2^3 = 8$  és ezek megfeleltethetők a  $3D$  egységkocka csúcsainak.



Kérdés: hogyan döntjük el, hogy a téves 001, 010, 100, 110, 101, 011 kódokat melyik jó (000 vagy 111) kódba dekódoljuk? Hova „lökjük/lökhetjük” a téves „pontokat”? Hány „lépéssel” lkjük őket a helyükre?



100  
Kérdés: mekkora a távolsága a 000 kódtól? Mely kódok vannak még ilyen távolságra?

011  
Kérdés: mekkora a távolsága az 111 kódtól? Mely kódok vannak még ilyen távolságra?

**Tétel (Plotkin korlát):** Bármely blokk kódra teljesül, hogy

$$M \leq 2^{n-2d+2}d.$$

A megfelelő blokk kód létezéséhez a Hamming és Plotkin korlátok csak szükséges, de nem elégséges feltételeket adnak. Ezért őket gyakran felső korlátoknak nevezzük. Léteznek elégséges feltételek (alsó korlátok) is egy kód létezésére adott  $M$ ,  $n$  és  $d$  esetén.

**Tétel (Gilbert korlát):** Ha a paraméterek kielégítik a

$$(M-1) \sum_{i=0}^{d-1} \binom{n}{i} < 2^n \quad (3.5)$$

egyenlőtlenséget, akkor létezik ilyen tulajdonságú blokk kód.

Ha  $k = \log M$  egész szám, akkor egy sokkal erősebb alsó korlát is ismert.

**Tétel (Varshamov-Gilbert korlát):** Ha a paraméterek kielégítik a

$$M \sum_{i=0}^{d-2} \binom{n-1}{i} < 2^n \quad (3.6)$$

egyenlőtlenséget, akkor létezik ilyen tulajdonságú blokk kód.

Példa: Lássuk az eredeti (2 hosszúságú) kódok bővítését 5 hosszúságúra ( $n=5$ ,  $k=2$  a korábbi jelölés szerint). Túlzás ez a bővítés a 3 hosszúságú redundánс részzel?

$$\begin{aligned}(0\ 0) &\rightarrow (0\ 0\ 0\ 0\ 0) \\(0\ 1) &\rightarrow (0\ 1\ 1\ 1\ 0) \\(1\ 0) &\rightarrow (1\ 0\ 1\ 0\ 1) \\(1\ 1) &\rightarrow (1\ 1\ 0\ 1\ 1)\end{aligned}$$

Legyen K az 1. példabeli kód,  $u=(0\ 0\ 0\ 0\ 0)$ ,  $t=1$ .

Az  $u$  körüli 1 sugarú gömbben a következő sorozatok vannak.

( 0 0 0 0 0 )
( 1 0 0 0 0 )
( 0 1 0 0 0 )
( 0 0 1 0 0 )
( 0 0 0 1 0 )
( 0 0 0 0 1 )

Ezeket a kódokat mindenkorban a  
00000  
kódszóként értelmezzük, ami  
egyértelműen a  
00  
eredeti üzenetet jelenti

Figyeljük meg, hogy nincs a gömb elemei között a  $(0\ 0\ 0\ 0\ 0)$ -t leszámítva kódszó, ami összhangban van azzal, hogy a kód távolsága nagyobb 1-nél (3).

Egy példa, ha hosszabb a kódszó (csak az elméleti korlátok figyelembevételével):

Állapítsuk meg, hogy van-e 5 minimális távolságú 13 hosszú perfekt bináris kód?

$n=13$ ,  $d=6$ , és  $\left\lfloor \frac{d-1}{2} \right\rfloor = 2$  így a kód  $t=2$ -hibajavító.

Ha a közleményszavak k hosszúak, akkor a közleményszavak száma  $2^k$ , ami megegyezik a kódszavak számával,  $M=2^k$ .

Ha a kód perfekt, akkor a Hamming-korlát egyenlőséggel teljesül:

$$2^k \left( \binom{13}{0} + \binom{13}{1} + \binom{13}{2} \right) = 2^{13}$$
$$2^k \left( 1 + 13 + \frac{12 \cdot 13}{2} \right) = 2^{13}$$
$$2^k \cdot 92 = 2^{13}$$

A jobb oldalon 2 hatványa áll, a bal oldalon szereplő 92 azonban nem 2 hatványa, ez az egyenlőség nem teljesülhet semmilyen k esetén. Ezért ilyen kód nem létezik

# ÁLTALÁNOSAN MEGÁLLAPÍTHATJUK, HOGY:

Shannon tétele alapján egy eléggé hosszú hibajavító kóddal tetszőlegesen kis hibavalószínűség érhető el, ha a kódolt bitek átviteli sebessége kisebb mint a csatorna kapacitása.

A hosszabb kóddal javul a hibajavító képessége, de nő a dekódolás bonyolultsága és ideje.

# VÉGEZETÜL:

- További példákat találhatnak Galántai tanár úr jegyzetében
- És a következő honlapokon:
- [http://home.mit.bme.hu/~benes/oktatas/dig-jegyz\\_052/kodolas.pdf](http://home.mit.bme.hu/~benes/oktatas/dig-jegyz_052/kodolas.pdf)
- <https://tudasbazis.sulinet.hu/hu/szakkepzes/elektronika-elektrotechnika/digitalis-alaparamkorok/digitalis-adatok-ellenorzese-es-javitasa/a-digitalis-adatok-ellenorzese-es-javitasa>
- <https://gyires.inf.unideb.hu/GyBITT/30/ch03s02.html>
- A következő oldalon találják a megoldásra váró feladatot, amelynek megoldását a Moodle rendszerbe várom (határidő a következő óra előtti nap, este 8 óra).
- Jó munkát!
-

- 1. Egy általános blokk-kódoló eljárás paramétereit definiálva (Hamming távolság, hibák száma) adja meg az ezek közötti alapvető összefüggést! Adjon magyarázatot az összefüggésre a Hamming gömbök segítségével!
- 2. Adott egy kódrendszer, mely négy-hat kódszóból áll. Keressük meg a minimális Hamming távolság ( $d$ ) értékét, állapítsuk meg, hány hibát tud jelezni, illetve hány hiba javítását teszi lehetővé a kódrendszer.. (pl: a kódszavak lehetnek: a, 0011111 b, 0011110 c, 0100101 d, 0100011, de javaslom, hogy egy másik példát állítson fel.).
- 3. Legyen  $M$  a teljes kódhalmaz,  $n$  a kódszavak hossza,  $d$  a kódtávolság. Mit jelent a perfekt kódok Hamming korlátja ezen paraméterek ismeretében? (Válaszát saját szavaival írja le!)
- 4. Adjon meg egy gyakorlati példát és magyarázatot erre a korlátra a fenti paraméterek Ön által megadott értékeire! (a 20. dián látható példa mintájára).
- Válaszait kézzel írottan, fotózva, vagy esszébe foglalva töltse fel a Moodle rendszerbe!

# Hibakorlátozó kódolás

7. Tematika

Forrás:

<http://tel.tmit.bme.hu/hirtech/Jegyzet/>

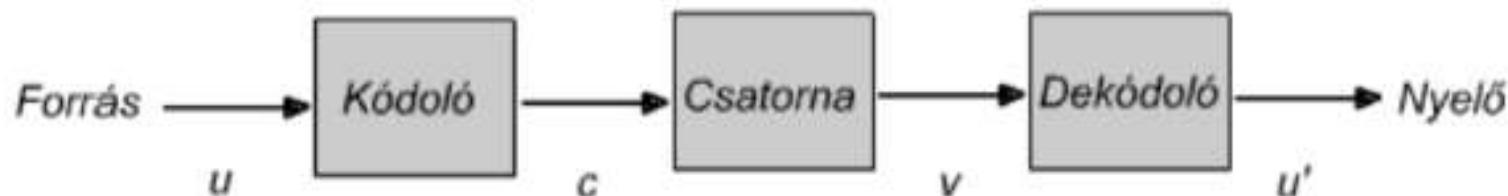
7. fejezet

# Miért?

- Forráskód \_ redundancia eltávolítás
- Csatornakód \_ kód+redundáns rész, a rekonstruálás biztonsága érdekében
- Zaj\_ a modulátor-demodulátor vonalon
- Ismétlés elkerülése – csatornakapacitás

A *hibakorlátozó kódolás* két feladata a *hibajelzés* és a *hibajavítás*. A hibajelzés folyamata az, hogy a vevő a hibajelző kód segítségével detektálja a vételi hibát, majd értesíti az adót egy visszairányú csatornán a detektált vételi hibáról, amely erre legtöbbször újraadást kezdeményez. Hibajavításon azt értjük, hogy a hibajavító kódolás alapján a vevő alkalmassá válik bizonyos meghibásodási esetek javítására. Gyakoriak a hibrid kódolási eljárások, amelynél a vevő először hibajavítást végez, majd a javítást ellenőrzi

# Alapfogalmak



7.1. ábra. Hibajavítás a hírközlési láncban

A forrás  $k$  hosszúságú

$$\mathbf{u} = (u_1, u_2, \dots, u_k)$$

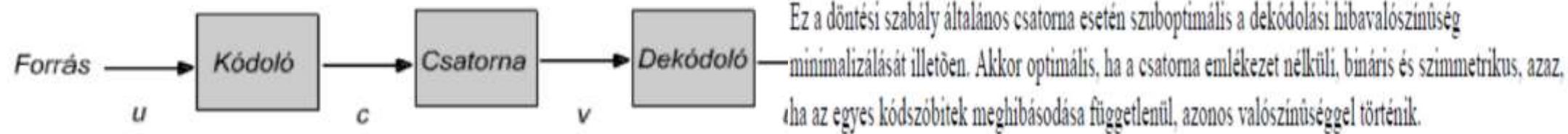
bináris üzenetet - melyet tekintsünk egyetlen forrásszónak - kíván eljuttatni hírközlési csatornán keresztül egy nyelőbe. A kódoló az üzenetet egy  $n \geq k$  hosszúságú

$$\mathbf{c} = (c_1, c_2, \dots, c_n)$$

bináris kódszóba képezi le. A csatorna kimenetén egy  $n$  hosszúságú

$$\mathbf{v} = (v_1, v_2, \dots, v_n)$$

bináris vett szó jelenik meg.



Azt mondjuk, hogy a csatorna az  $m$ -edik pozícióban hibázott, ha  $v_m \geq c_m$ . Jelölje  $t$  az összes ilyen hibázások számát  $\mathbf{c}$  átvitelekor. Általában, tetszőleges  $\mathbf{c}$  és  $\mathbf{v}$  szavak *Hamming-távolsága* -  $d(\mathbf{c}, \mathbf{v})$  - azon pozíciók száma, amelyben azok különböznek. Tehát

$$t = d(\mathbf{c}, \mathbf{v}).$$

*Kódon* (blokk-kódon)  $n$  hosszúságú vektorok  $C$  részhalmazát értjük, amelynek mérete  $k$  hosszúságú bináris üzenetek esetén  $2^k$ . Gyakran  $C(n, k)$  alakban jelöljük a kódot, ahol  $n$  és  $k$  a kód paraméterei. A kód elemei a *kódszavak*. A kódolás kölcsönösen egyértelmű leképezés, amely az egyes üzeneteket kódszavakra képezi le, tehát különböző üzenetek különböző kódszavakra képződnek le.

*Dekódoláson* ebben a fejezetben két lépés egymás utáni végrehajtását értjük. Először a  $\mathbf{v}$  vett szót egy *dekódolási döntési szabály* alapján a  $\mathbf{c}'$  dekódolt kódszóba képezzük le, majd a kódolás inverzének megfelelően ehhez a  $\mathbf{c}'$  kódszóhoz egy  $\mathbf{u}'$  üzenetet rendelünk.

A leggyakrabban alkalmazott dekódolási döntési szabály az, hogy a  $\mathbf{v}$  vett szóhoz Hamming-távolságban legközelebbi  $\mathbf{c}'$  kódszót kell választani, azaz

$$d(\mathbf{c}', \mathbf{v}) = \min_{\mathbf{c} \in C} d(\mathbf{c}, \mathbf{v}).$$

Az eddigiekből következik, hogy két fő feladatot kell elvégezni a hibajavító kódolással kapcsolatban.

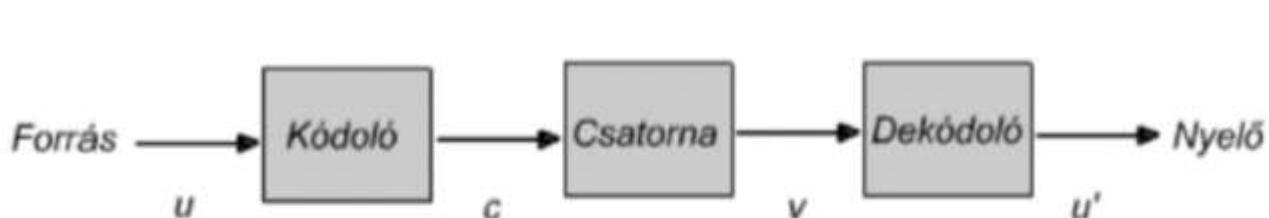
Konstruálni kell olyan kódot, amelyben a kódszavak közötti Hamming-távolság a lehető legnagyobb. Ha adott egy kód, olyan dekódolási döntési szabályt kell konstruálni, hogy a vett szóhoz minimális távolságra levő kódszó megtalálásához ne kelljen végigvizsgálni az összes kódszót. Kisméretű kódok esetén a végigvizsgálás még lehetséges. Ha a kódszóhossz, azaz a kód  $n$  paramétere viszonyság kicsi (pl.  $n < 10$ )

akkor ún. *táblázatos dekódolást* végezhetünk. A táblázatban az dekódolt kódszót, valamint az ahhoz tartozó üzenetet. (Gyakor dekódolt üzenetet adjuk meg.) Az eljárás szemléltetéséül tekin-

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

u	c
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

A 32 soros dekódolási táblázat első tíz sora a következő:



v	c'	u'
0 0 0 0 0	0 0 0 0 0	0 0
1 0 0 0 0	0 0 0 0 0	0 0
0 1 0 0 0	0 0 0 0 0	0 0
1 1 0 0 0	0 0 0 0 0	0 0
0 0 1 0 0	0 0 0 0 0	0 0
1 0 1 0 0	1 0 1 1 0	1 0
0 1 1 0 0	0 1 1 0 1	0 1
1 1 1 0 0	0 1 1 0 1	0 1
0 0 0 1 0	0 0 0 0 0	0 0
1 0 0 1 0	1 0 1 1 0	1 0

Általában azonban a tárkapacitás nem elég nagy a táblázatos dekódoláshoz. Ha viszont  $k$  kicsi értékű ( $k \ll n$ ), akkor még nagy kódszóhossz esetén is csak kevés kódszó van, és ekkor kódszavanként külön-külön ki lehet számítani a távolságot - gyakorlatilag megvalósítható dekódolóval. Tipikus kódparaméterek esetén azonban egyik út sem járható. Gondoljunk arra, hogy pl.  $k = 50$  bit üzenethosszúság esetén  $2^{50} \sim 10^{15}$  méretű a kód! A megoldás megértéséhez további alapfogalmakat kell megismernünk.

Egy kód kódszavai közötti minimális Hamming-távolság a kód igen fontos jellemzője, amelyet **kódtávolságnak** nevezünk és  $d_{\min}$ -nel jelölünk. Tehát formálisan:

$$d_{\min} = \min_{\substack{\mathbf{c} \neq \mathbf{c}' \\ \mathbf{c}, \mathbf{c}' \in C}} d(\mathbf{c}, \mathbf{c}').$$

Könnyen ellenőrizhető, hogy a 7.1. példában  $d_{\min} = 3$ .

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

<b>u</b>	<b>c</b>
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

A 32 soros dekódolási táblázat első tíz sora a következő:

<b>v</b>	<b>c'</b>	<b>u'</b>
0 0 0 0 0	0 0 0 0 0	0 0
1 0 0 0 0	0 0 0 0 0	0 0

*Hibajelzés* során a feladat annak megállapítása, hogy a vett szó kódszó-e vagy sem. Az átküldött kódszó bitjei a hibák helyén invertálva jelennek meg a vett szóban. Ha a hibák száma legfeljebb  $t$  lehet egy kódszó vétele során, továbbá, ha

$$d_{min} > t,$$

akkor biztosak lehetünk abban, hogy hiba esetén nem keletkezhet téves kódszó. Ha ugyanis olyan meghibásodás állna elő, amivel a vett szó éppen téves kódszó lenne, ezt a meghibásodást vételi oldalon nem tudnánk jelezni, azaz *detektálási hiba* keletkezne. Következésképpen

*7.1. téTEL:* Egy  $d_{min}$  kódtávolságú  $C$  kód minden, legfeljebb  $d_{min}-1$  számú hibát jelezni tud.

*Hibajavítás* esetén azt kérdezzük, hogy ha  $t$  a hibák száma, akkor mi biztosítja azt, hogy a  $v$  vett szóból a  $c$  küldött kódszó egyértelműen visszaállítható legyen. Ennek a formális feltétele az, hogy minden más  $c'$  kódszóra fennálljon a

$$d(v, c') > d(v, c) \quad (7.1)$$

egyenlőtlenség, azaz egyértelműen  $c$  legyen legközelebb a vett szóhoz. Mivel a Hamming-távolság valóban távolság (nemnegatív, szimmetrikus és teljesíti a háromszög-egyenlőtlenséget), ezért a háromszög-egyenlőtlenség alapján

$$d(v, c') \geq d(c, c') - d(v, c), \quad (7.2)$$

így a (7.1) cél elérhető, ha a (7.2) jobb oldala nagyobb, mint  $d(v, c)$ , azaz

$$d(c, c') - d(v, c) > d(v, c).$$

Innen a  $d(c, c') > 2d(v, c)$  átrendezett alakot kapjuk. Ez az egyenlőtlenség biztosan teljesül, ha a kódtávolságból adódó  $d(c, c') \geq d_{min}$ ,  $c \neq c'$  összefüggést is figyelembe véve a

$$d_{min}/2 > d(v, c)$$

egyenlőtlenség fennáll. Összefoglalva:

*7.2. téTEL:* Egy  $d$  kódtávolságú  $C$  kód hibajavító képessége  $\text{int}[(d_{min} - 1)/2]$ .

A 7.1. példabeli kód hibajavító képessége 1, vagyis ha egyetlen bithiba van, legyen az bármelyik kódszó pozícióján, a kód azt javítani tudja. A 7.2. tételeből következik, hogy ha egy kód minden, legfeljebb  $t$  hibát javítani képes, akkor  $d_{min} > 2t+1$  áll fenn a kódtávolságára.

Felvetődik ezek után a kérdés: hogyan konstruálhatunk elegendően nagy kódtávolságú kódokat? Ezzel kapcsolatosan a következő pontban egy fontos kódosztály, a lineáris kódok alapfogalmait tekintjük át. A lineáris kódok előnye, hogy matematikai leírásmódjuk egyszerűbb, számos kódkonstrukció és hatékony dekódolási eljárás létezik erre a kódosztályra, továbbá implementálásuk is egyszerűbb, mint az általános nemlineáris kódoké.

*7.1. példa.* Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

<b>u</b>	<b>c</b>
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

A 32 soros dekódolási táblázat első tíz sora a következő:

<b>v</b>	<b>c'</b>	<b>u'</b>
0 0 0 0 0	0 0 0 0 0	0 0
1 0 0 0 0	0 0 0 0 0	0 0
0 1 0 0 0	0 0 0 0 0	0 0
1 1 0 0 0	0 0 0 0 0	0 0
0 0 1 0 0	0 0 0 0 0	0 0
1 0 1 0 0	1 0 1 1 0	1 0
0 1 1 0 0	0 1 1 0 1	0 1
1 1 1 0 0	0 1 1 0 0	0 0
0 0 0 1 0	0 0 0 0 1	0 1
0 1 0 0 1	0 0 0 1 0	1 0

# Lineáris kódok

7.2. példa. Tekintsük a 7.1. példa kódját. Észrevehetjük, hogy a kódolást elvégezhetjük egy  $\mathbf{c} = \mathbf{u}\mathbf{G}$  mátrixszorzással, ahol  $\mathbf{G}$  egy

$$\mathbf{G} = \begin{bmatrix} 10110 \\ 01101 \end{bmatrix}$$

alakú,  $2 \times 5$  méretű bináris mátrix. Ebben az egyszerű esetben ez azt jelenti, hogy a kódszavak halmazát a csupa zérus vektor, a  $\mathbf{G}$  mátrix első, ill. második sora, valamint két sorának koordinátánkénti modulo 2 (XOR) összege adja. Tehát a  $\mathbf{G}$  sorainak lineáris kombinációi képezik a  $C$  kód elemeit. A  $C$  kód mint bináris vektorok halmaza tehát algebrai struktúrát, lineáris teret alkot. Ezt az észrevételt általánosítva jutunk el a lineáris kódok fogalmához.

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

<b>u</b>	<b>c</b>
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

Egy  $C$  bináris kódot *lineáris kódnak* nevezünk, ha a  $C$  halmaz *lineáris tér*, azaz ha minden  $\mathbf{c}, \mathbf{c}' \in C$  esetén  $\mathbf{c} + \mathbf{c}' \in C$  is fennáll. Innen következik, hogy a csupa zérus  $\mathbf{0}$  szó is eleme kell, hogy legyen a lineáris kódnak, hiszen tetszőleges  $\mathbf{c}$  bináris szó esetén fennáll a  $\mathbf{c} + \mathbf{c} = \mathbf{0}$  egyenlőség. A lineáris kódok jelentőségét az adja, hogy az egyes üzenetekhez tartozó kódszavak viszonylag egyszerűen generálhatók, valamint egyszerűbb a hibadetektálás és a hibajavítás.

A valós vektortérben megszokott fogalmak a bináris vektorok terében is érvényesek. Ennek megfelelően alkossák a  $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$   $C$ -beli vektorok a  $2^k$  elemet tartalmazó  $C$  lineáris tér egy bázisát, azaz ezen vektorok segítségével egyértelműen előállíthatunk tetszőleges  $\mathbf{c} \in C$  elemet a

$$\mathbf{c} = \sum_{i=1}^k u_i \mathbf{g}_i$$

alakban,  $u_i$  bináris együtthatókkal,  $i = 1, 2, \dots, k$ . Képezzünk egy  $\mathbf{G}$   $k \times n$  méretű mátrixot, amelynek a  $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$  vektorok a sorai. A kódolás ezek után a

$$\mathbf{c} = \mathbf{u}\mathbf{G} \tag{7.3}$$

művelettel elvégezhető. A  $\mathbf{G}$  mátrixot - kézenfekvően - *generátmátrixnak* nevezzük.

A  $C$  kódot tehát tömören megadhatjuk -  $k$  megfelelően kiválasztott kódszavával - , és nem szükséges felsorolni valamennyi, azaz  $2^k$  kódszavát. Továbbá a kódolás is egyszerű szabály szerint történik. Vegyük észre, hogy egy kódhoz számíthatan generátor mátrix tartozik, azaz ahány különböző bázisa lehet egy lineáris térnek. Viszont egy adott kódoláshoz, azaz üzenet - kódszó összerendeléshez csak egy adott generátor mátrix tartozik.

Ha megnézzük a 7.1. példabeli kódunkat, észrevehetjük, hogy a kódolást úgy választottuk, hogy a kódszavak első két bitje azonos legyen a megfelelő üzenet két bitjével. Ez előnyös, mivel a dekódolás második lépése, az üzenet-hozzárendelés a dekódolt kódszóhoz triviális, hiszen ez egyszerűen a dekódolt kódszó első  $k$  bitjének leválasztását jelenti. Ezt a kódolást más esetben is megtehetjük.

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

<b>u</b>	<b>c</b>
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

Egy  $C(n,k)$  kód *szisztematikus*, ha kódszavának első  $k$  bitje megfelel az üzenetnek. Szisztematikus kód esetén már egyértelmű a generátor mátrix, és a mátrix-szorzás szabályai alapján nyilvánvalóan a következő alakú:

$$\mathbf{G} = (\mathbf{I}_k, \mathbf{B}), \quad (7.4)$$

ahol  $\mathbf{I}_k$  egy  $k \times k$  méretű egységmátrix,  $\mathbf{B}$  pedig egy  $k \times (n - k)$  méretű mátrix. Az  $\mathbf{u}$  üzenethez tartozó kódszó felépítése tehát a következő:

$$\mathbf{c} = (u_1, u_2, \dots, u_k, c_{k+1}, c_{k+2}, \dots, c_n).$$

A  $\mathbf{c}$  első  $k$  koordinátájából álló szegmensét *üzenetszegmensnek*, az utolsó  $n - k$  koordinátájából álló szegmensét pedig *paritásszegmensnek* nevezük.

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

$\mathbf{u}$	$\mathbf{c}$
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

Egy  $C$  lineáris kódhoz hozzárendelhetjük a  $\mathbf{H}$ ,  $(n - k) \times n$  méretű bináris mátrixot, amelynek az a tulajdonsága, hogy detektálni tudja az  $n$  hosszúságú bináris vektorok  $2^n$  méretű halmazában a  $C$  kódszavait. A detektálás alapja az, hogy a

$$\mathbf{H}\mathbf{c}^T = 0 \quad (7.5)$$

összefüggés akkor és csak akkor áll fenn, ha  $\mathbf{c} \in C$  ( $T$  a transzponálás jele). Az ilyen tulajdonságú  $\mathbf{H}$  mátrixot *paritás-ellenőrző mátrixnak* nevezzük. Szisztematikus generálású kódok esetén egyszerűen megkaphatjuk a  $\mathbf{H}$  mátrixot:

$$\mathbf{H} = (\mathbf{A}, \mathbf{I}_{n-k}), \quad (7.6)$$

ahol

$$\mathbf{A} = -\mathbf{B}^T, \quad (7.7)$$

továbbá  $\mathbf{I}_{n-k}$   $(n - k) \times (n - k)$  méretű egységmátrix. A (7.6), (7.7) állítások egyszerűen igazolhatók: (7.3) és (7.5)-ből a tetszőleges összetartozó  $\mathbf{c}, \mathbf{u}$  párra a

$$\mathbf{H}\mathbf{c}^T = \mathbf{H}(\mathbf{u}\mathbf{G})^T = \mathbf{H}\mathbf{G}^T\mathbf{u}^T = 0$$

egyenlőségláncot kapjuk, ahonnan

$$\mathbf{H}\mathbf{G}^T = 0 \quad (7.8)$$

összefüggés adódik. A (7.8) egyenlőségbe behelyettesítve a (7.4) valamint a (7.6) összefüggéseket:

$$\mathbf{H}\mathbf{G}^T = (\mathbf{A}, \mathbf{I}_{n-k})(\mathbf{I}_k, \mathbf{B})^T = \mathbf{A} + \mathbf{B}^T = 0$$

alapján (7.7) igazolására jutottunk.

7.3. példa. Tekintsük a 7.1. példa  $C$  kódját, és annak 7.2. példabeli generátor mátrixát. A bevezetett jelölésekkel használva:

$$\mathbf{B} = \begin{bmatrix} 110 \\ 101 \end{bmatrix}$$

7.1. példa. Tekintsük az alábbi kódolás szerinti  $C(5,2)$  kódot:

$\mathbf{u}$	$\mathbf{c}$
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

ahonnan  $-1 = 1$  (modulo 2) figyelembevételével

$$\mathbf{A} = \begin{bmatrix} 11 \\ 10 \\ 01 \end{bmatrix}$$

adódik. Tehát a keresett  $\mathbf{H}$  paritásmátrix:

$$\mathbf{H} = \begin{bmatrix} 11100 \\ 10010 \\ 01001 \end{bmatrix}$$

A következőkben azt mutatjuk meg, hogyan használható a  $\mathbf{H}$  mátrix a dekódolásnál.

$$\mathbf{G} = \begin{bmatrix} 10110 \\ 01101 \end{bmatrix}$$

$$\mathbf{G} = (\mathbf{I}_k, \mathbf{B}),$$

Legyen az átküldött kódszó  $\mathbf{c}$ , a vett szó  $\mathbf{v}$ . Az  $\mathbf{e} = \mathbf{v} - \mathbf{c}$  vektort *hibavektornak* nevezzük.

Például, ha  $\mathbf{c} = (10110)$ , a vett szó pedig  $\mathbf{v} = (11110)$ , akkor a hibavektor  $\mathbf{e} = (01000)$ , azaz a 2. koordináta hibásodott meg.

$$\mathbf{H} = (\mathbf{A}, \mathbf{I}_{n-k}),$$

$$\mathbf{A} = -\mathbf{B}^T,$$

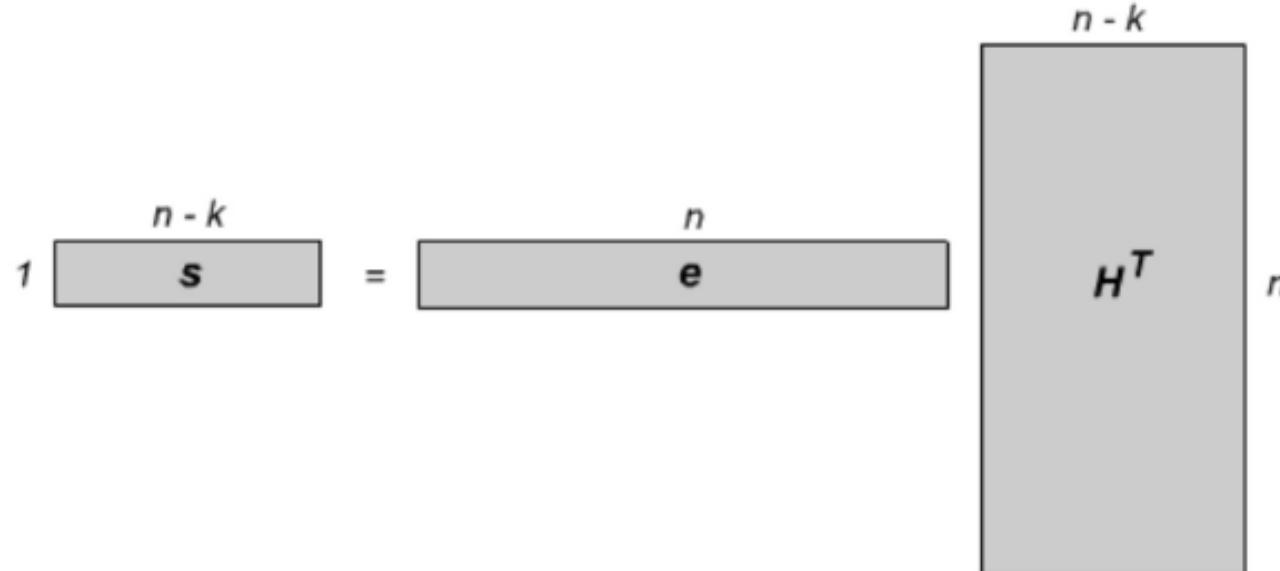
Vegyük észre, hogy (7.5) felhasználásával

$$\mathbf{Hv}^T = \mathbf{H}(\mathbf{c} + \mathbf{e})^T = \mathbf{H}\mathbf{c}^T + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T,$$

vagyis  $\mathbf{Hv}^T$  értéke csak a hibavektortól függ, az adott kódszótól nem. Az

$$s = \mathbf{eH}^T$$

mennyiséget az  $\mathbf{e}$  hibavektor *szindrómájának* nevezzük. A kódszavak szindrómája 0. (A  $\mathbf{H}\mathbf{e}^T$  oszlopvektornak megfelelő sorvektor  $\mathbf{eH}^T$ .) Az  $s = \mathbf{eH}^T$  szorzás szemléltetését láthatjuk az 7.2. ábrán.



7.2. ábra. A szindrómaszámítás dimenziójának szemléltetése

Például a 7.3. példabeli  $\mathbf{H}$  mátrix esetén az  $\mathbf{e} = (01000)$  hibavektor szindrómája  $s = (101)$ . A szindrómavektor hossza  $n - k$ , azaz esetünkben  $5 - 2 = 3$ .

A szindróma felhasználásával a lineáris kódok egy táblázatos dekódolása adható meg. A dekódolási táblázat felépítése a következő:

szindróma	min. hibaszámú hibavektor
$s_0=0$	$e_0=0$
$s_1$	$e_1$
.	.
.	.
.	.
$s_{2n-k-1}$	$e_{2n-k-1}$

A dekódolási táblázat két oszlopa közül az elsőben a szindrómák, a másodikban a szindrómáknak megfelelő, minimális hibaszámú hibavektorok állnak. A nulladik sorban a zérus szindróma és a neki megfelelő zérus hibaminta áll, ami a hibamentes esetnek felel meg. A szindrómavezektorok hossza  $n - k$ , így a különböző szindrómák száma  $2^{n-k}$ .

Ezek után a *szindrómadekódolás* lépései a következők:

1. A  $v$  vett szónak megfelelő  $s$  szindróma kiszámítása.
2. A dekódolási táblázat  $s$ -nek megfelelő sorából a becsült  $e$  hibavektor kiolvasása.
3. A  $c' = v - e$  dekódolási lépés elvégzése.
4. Az  $u'$  dekódolt üzenet  $c'$ -hez rendelése.



Tekintsük a következő példát!

7.4. példa. Az 7.3. példabeli  $\mathbf{H}$  paritásmátrix felhasználásával a  $C(5, 2)$  kódunk esetén a szindrómádekódolási táblázat a következő alakú:

s	e
0 0 0	0 0 0 0 0
1 0 0	0 0 1 0 0
0 1 0	0 0 0 1 0
0 0 1	0 0 0 0 1
1 0 1	0 1 0 0 0
0 1 1	0 0 0 1 1
1 1 1	0 1 0 1 0

A táblázatra tekintve észrevehetjük, hogy a kód az 5 különböző egyes hibát, valamint egyetlen kettős hibát képes javítani.

Példák:

### 7.3.1. Ismétléses kód

A legegyszerűbb hibajavító kód az ismétléses kód. Ekkor az üzenet  $k = 1$  méretű, és ezt ismétljük meg  $n$ -szer. Így egy  $C(n,1)$  kódot kapunk. A két különböző (0, ill. 1) üzenetnek megfelelően ez az egyszerű kód két kódszót tartalmaz a (000...0) és az (111...1) kódszavakat. A kódtávolság nyilván  $n$ , így a kód hibajavító képessége, ha  $n$ -et célszerűen páratlan értékűre választjuk,  $(n-1)/2$ .

7.5. példa. A (000) és (111) szavakat tartalmazó kód a legegyszerűbb 1 hibát javító kód. Így pl. ha a vett szó (110), akkor a (111) kódszót dekódoljuk, és az 1 üzenetre döntünk.

### 7.3.2. Egyszerû paritáskód

A legegyszerûbb hibadetektáló kód az egyszerû paritáskód. Egy  $\mathbf{u}$  üzenethez az üzenet bitjeinek megfelelő páros vagy páratlan paritásbitet illesztve kapjuk a kódszót (a páros paritás az elemek modulo 2 összege, a páratlan annak inverze). Így a paritáskód  $C(n, n-1)$  paraméterû. Könnyen belátható, hogy a kódtávolság 2. Ugyanis, ha tetszőleges kódban egy bitet tetszőleges koordinátán megváltoztatunk, megváltozik a paritás. Két bitet változtatva visszakapjuk az eredeti (páros vagy páratlan) paritást, tehát újra egy érvényes kódszóra jutunk, amely az eredeti kódszótól két koordinátában különbözik. Mivel a kódtávolság 2, ezért a 7.1. téTEL szerint egy hiba detektálására alkalmas az egyszerû paritáskód.

### 7.3.3. Kétdimenziós paritáskód

Rendezzük a  $k$  üzenetbitet egy  $\mathbf{U} p \times q$  méretű mátrixba, azaz  $k = pq$ . Képezzünk soronként, ill. oszloponként paritást, és az így adódó paritásbiteket írjuk a sor következő elemeként a sor végére, ill. az oszlop alá (7.3. ábra). Ekkor egy  $\mathbf{C} (p+1) \times (q+1)$  méretű mátrixot kaphatunk, ha a mátrix még nem definiált jobb alsó sarokelemét is megadjuk. Legyen ez a sarokelem a "paritások paritása", azaz a  $\mathbf{C}$  mátrix utolsó sorában vagy utolsó oszlopában álló paritásbíték paritása (könnyen látható, hogy minden egyik úton azonos érték kerül a jobb alsó sarokba).

A kódtávolság 4, amit a következőképp láthatunk be. Tekintsük a páros paritású esetet. Tartalmazzon az  $\mathbf{U}$  üzenetmátrix egyetlen 1-et. Ekkor az 1-nek megfelelő sorban, ill. oszlopban a paritásbit 1, továbbá a jobb alsó sarokba kerülő paritásbit is 1. Tehát ezen kódszóban 4 db. 1 lesz, azaz a csupa zérus kódszótól való távolsága 4 (a Hamming-távolságot a mátrixok megfelelő koordinátáinak egybevetésével nyerjük). Tetszőleges kód esetén két tetszőleges kódszó Hamming-távolsága nyilván azonos a különbségükben található nemzérus elemek számával. Továbbá lineáris kódok esetén a kódszavak különbsége is kódszó. Következetképpen a kódszópárok közötti minimális távolság azonos a legkevesebb 1-et tartalmazó nemzérus kódszó 1-einek a számával. Az esetek végigpróbálásával könnyen belátható, hogy a kétdimenziós paritáskód esetén 4-nél kevesebb 1-et tartalmazó nemzérus kódszó nincsen, így a a kódtávolság valóban 4.

		$q + 1$
	$p$	$q$
$P + 1$	$p$	1
1		□

7.3. ábra. Kétdimenziós paritáskód

### 7.3.4. Hamming kód

A szindrómaszámítás  $s = \mathbf{eH}^T$  definíciójára tekintve láthatjuk (7.2. ábra), hogy azon  $\mathbf{e}$  vektorhoz, amely  $(000\dots 1\dots 0)$  alakú, azaz egyetlen 1-et tartalmaz az  $i$ -edik koordinátáján, a  $\mathbf{H}$  mátrix  $i$ -edik oszlopa rendelődik szindrómaként. Következésképpen, ha azt szeretnénk, hogy az  $n$  különböző ilyen alakú hibavektorhoz különböző szindrómák tartozzanak, a  $\mathbf{H}$  oszlopait egymástól és nullától különbözőknek kell választani. Ez a választás tehát garantálja, hogy a  $\mathbf{H}$  mátrixnak megfelelő kód minden 1 hibát tartalmazó meghibásodást javítani tudjon. A  $\mathbf{H}$  mátrixnak megfelelő kódot, ill.  $\mathbf{G}$  generátor mátrixot úgy kaphatjuk meg egyszerűen, hogy a  $\mathbf{H}$  mátrixot szisztematikus (7.6) alakban építjük fel, majd a (7.7) összefüggést használjuk.

7.6. példa. Ezen konstrukciónak megfelelően egy 1 hibát javító  $C(7, 4)$  kód  $\mathbf{H}$  mátrixa lehet a következő:

$$\mathbf{H} = \begin{bmatrix} 1101100 \\ 1011010 \\ 0111001 \end{bmatrix}$$

Ennek a kódnak a hossza  $n = 2^3 - 1 = 7$ , így  $\mathbf{H}$  oszlopaiként a 7 különböző  $n-k = 7-4 = 3$  kódszóhosszú bináris nemzérus vektort választottuk. A  $\mathbf{H}$  mátrixnak megfeleltethető  $\mathbf{G}$  generátor mátrix a következő:

$$\mathbf{G} = \begin{bmatrix} 1000110 \\ 0100101 \\ 0010011 \\ 0001111 \end{bmatrix}$$

A megkonstruált kód egy  $(7, 4)$  paraméterű Hamming-kód. Észrevehetjük, hogy ez a kódkonstrukció egyfélre értelemben optimális. Nevezetesen, ha a feladat  $n = 7$  hosszúság mellett 1 hibát javító kódok konstruálása, akkor ezen kódok között nincsen olyan kód, amely  $2^4 = 16$ -nál nagyobb méretű.

Természetesen a konstrukció általánosítható ( $n = 2^r - 1$ ,  $k = 2^r - 1 - r$ ) paraméterek esetére, ahol  $r \geq 3$  egész szám.

- Matematikai alapok:
- Lásd Galántai tanár úr jegyzetét!

# Adattömörítés

8. óra

# Veszteségmentes tömörítés

## • Veszteségmentes tömörítés

- Futáshossz-kódolás
  - PackBits
  - RLE (a PCX használta például)
- Minimális redundanciájú kódolás
  - Huffman-kódolás (egyszerű entrópia kódolás)
  - Aritmetikai kódolás (felettebb entrópia kódolás)
- Lexikai kódolás
  - DEFLATE
  - LZ77 és LZ78
  - LZW
  - Más LZ tömörítési eljárások
- Burrows–Wheeler-transzformáció (blokkrendezési feldolgozás, amely a tömörítést egyszerűbbé teszi)

A run-length encoding egy nagyon egyszerű tömörítési eljárás, melyben az adatban található, hosszasan ismétlődő karaktereket egyetlen értékként és számként tárolják, az eredeti teljes karaktersorozat helyett. Ez leginkább sok ilyen hosszú karaktersorozattal rendelkező adatra hasznos: például egyszerű grafikus képek, mint ikonok, vonalrajzok és animációk. Nem hasznos azonban olyan adatokra, amelyeknél nincs sok ilyen karaktersorozat, hiszen jelentősen megnövelheti a fájlméretet.

MP3

JPEG

Fraktáltömörítés

Fraktálatalakítás

Hullámtömörítés

Gyakoriságok  
figyelembevétele

A következőkben a veszteségmentes adattömörítés alapfogalmaival foglalkozunk.

Adattömörítés: egy fájl méretének csökkentése.

Lehetséges céljai:

- Tárolóhely igény csökkentése.
- Adatátvitel idejének csökkentése.

Az adattömörítés lehetőségét az adja, hogy a legtöbb fájl redundanciákat tartalmaz.

**Példa:** Az angol nyelv információ tartalma:

	bit/karakter
bitek	7
entrópia	4.5
Huffman kód (átlag)	4.7
entrópia (8-as blokkok)	2.4
aszimptotika	1.3
Compress	3.7
Gzip	2.7
BOA	2.0

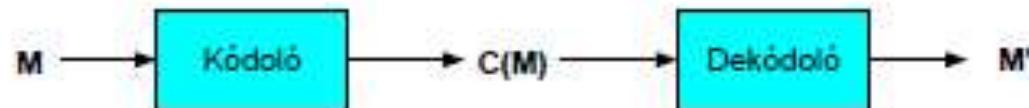
A táblázat utolsó három sora tömörítő programok teljesítményét jelzi.

# Alapfogalmak

**Üzenet:** Az  $M$  bináris adat(fájl), amelyet tömöríteni akarunk.

**Kódoló:** Az  $M$  egy "tömörített"  $C(M)$  reprezentációját állítja elő, amely remélhetően kevesebb bitet használ.

**Dekódoló:** Rekonstruálja az eredeti üzenetet, vagy annak egy  $M'$  közelítését.



**Kompresszió hányados:**

$$\frac{C(M) \text{ bitjeinek száma}}{M \text{ bitjeinek száma}}.$$

A tömörítés *veszteségmentes*, ha a dekódoló pontosan előállítja az eredeti  $M$  üzenetet. Egyébként pedig *veszteséges*.

A veszteségmentes tömörítésnél a tipikus kompresszió hányados: 50-75% vagy kevesebb.

**Példa:** Beszélt nyelvek, forráskódok, végrehajtható kódok.

A veszteséges tömörítésnél a tipikus kompresszió hányados: 10% vagy kevesebb.

**Példa:** Képek, hang, video.

A tömörítés elvi korlátai: Shannon elmélete. Kolmogorov komplexitás.

Itt most a következő "tételek" igazoljuk.

**Tétel:** Lehetetlen veszteségmentesen tömöríteni az összes fájlt.

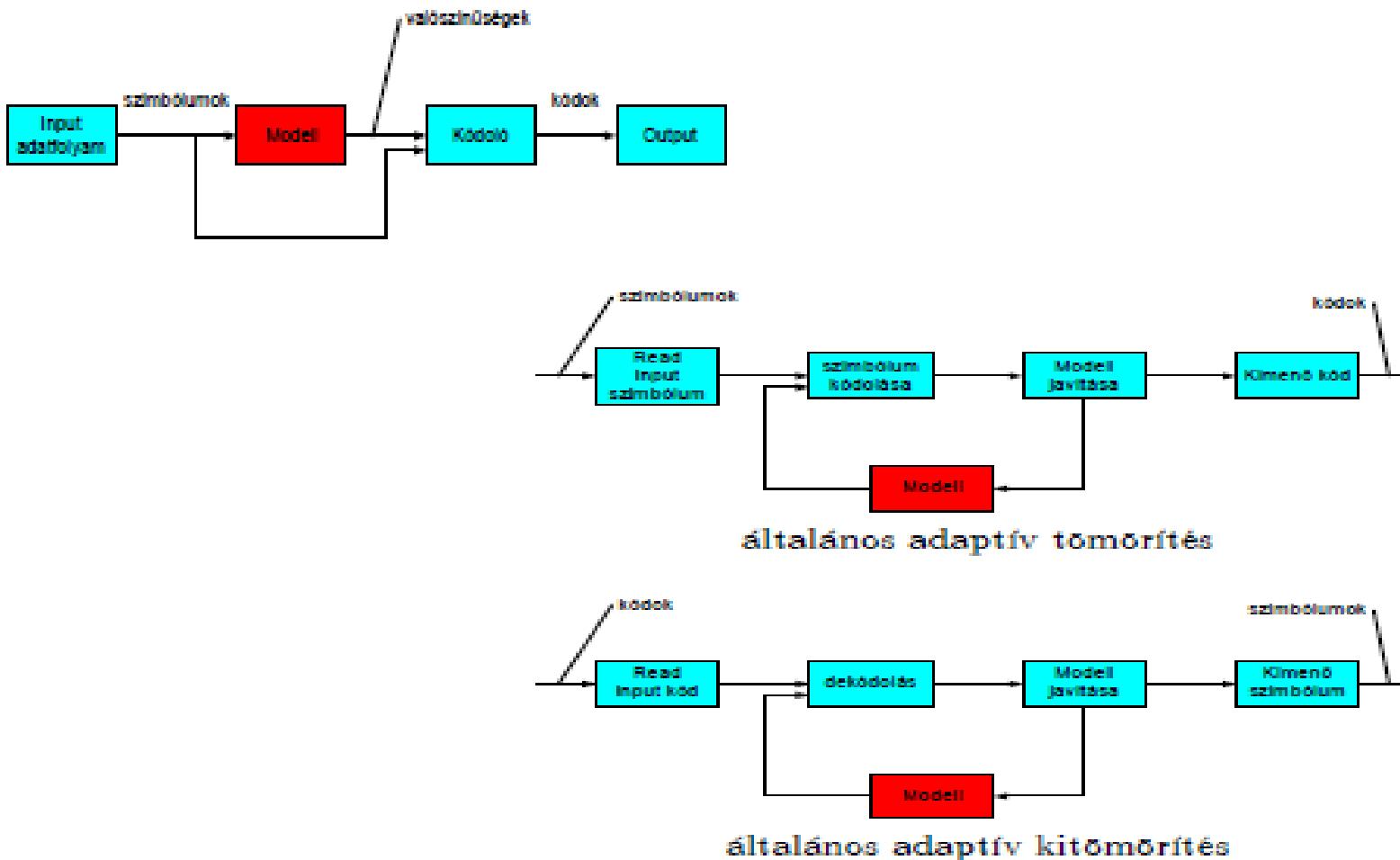
A tömörítő algoritmusoknak két fontos komponensük van:

- a modell,
- a kódoló.

Másképpen megfogalmazva:

$$\text{adattömörítés} = \text{modellezés} + \text{kódolás}$$

A modell komponens az üzenetek valószínűség eloszlását vagy valamilyen szerkezeti összefüggését adja meg (modellezzi). A kódoló ezt az ismeretet próbálja kihasználni.



Számos technika ismert a kódolási komponensre, de a legtöbb esetben ezek általános kódolók: a Huffman kód, vagy az aritmetikai kód.

A statisztikai modellek osztályozása:

- *Statikus modell*: Ugyanaz a modell minden szövegre.
  - Gyors.
  - Nem optimális: különböző szövegek különböző statisztikai tulajdonságokkal rendelkeznek.
  - Példa: ASCII, Morse kód.
- *Dinamikus modell*: a szövegen alapuló modellt generál.
  - Előfutás szükséges a modell generálásához.
  - A modellt is közvetíteni kell.
  - Példa: Huffman kód.
- *Adaptív modell*: Progresszíven tanulja és módosítja a modellt a szöveg olvasásával párhuzamosan.
  - A pontosabb modellezés jobb tömörítést eredményez.
  - A dekódolás a kezdettől kell, hogy induljon.
  - Példa: LZW.

# Futamhossz tömörítés

Ez egy egyszerű technika. Az RLE (Run-Length Encoding) jellemzői:

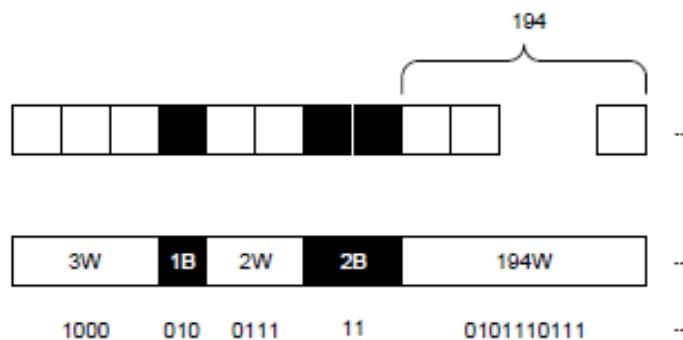
- Az ismétlődő karakterek hosszú futamait használja ki.
- Bináris ábécé esetén a futamok 0 és 1 között váltakoznak, a kód kimenete: a 0 vagy 1 jelek száma.
- fájl növekedés lehetséges, ha a futamok rövidek.

A futamhossz kódolás fontosabb alkalmazásai:

- JPEG,
- ITU-T T4 Group 3 fax (fekete-fehér grafika).

A Group 3 fax olyan képeket továbbít, amelyek soronként legfeljebb 1728 fekete vagy fehér pixelből (pixel=pel) állnak. A futamhosszakat a következő prefix kóddal (Huffmann kód) kódolja:

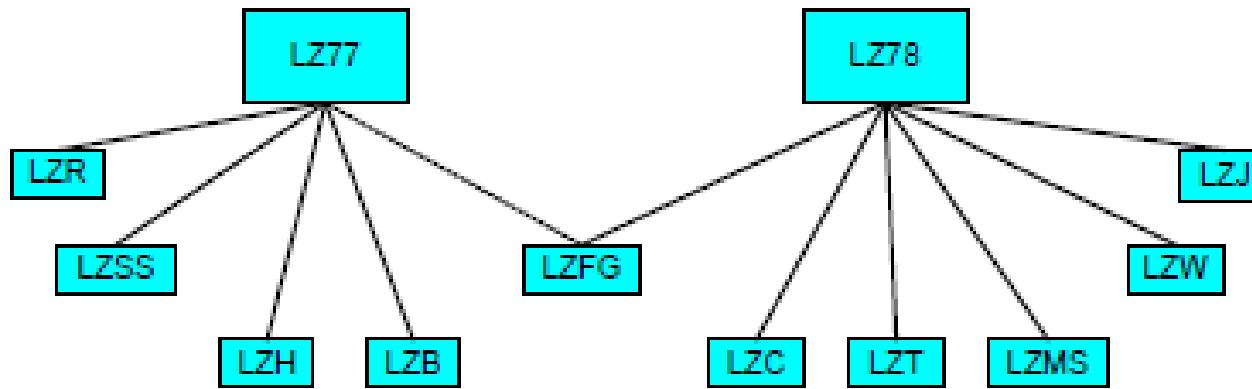
Példa:



futamhossz	white code	black code
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1011	011
...	...	...
20	0001000	00001101000
...	...	...
64+	11011	0000001111
128+	10010	000011001000

# LZV típusok

Az eredeti két algoritmust Lempel és Ziv javasolta 1977-ben és 1978-ban (LZ77, LZ78). Azóta a módosításokkal/variánsokkal együtt egy jelentős gyakorlati felhasználással rendelkező eljárás-  
saláddá vált:



Alkalmazások: zip, gzip, stacker, GIF, V.42, compress (LZ78), stb.

Az LZ tömörítő algoritmusok a szótár típusú kódolók közé tartoznak. A szótár típusú kódolók az adat részek közti korrelációk (ismétlődő minták) megfigyelésén alapulnak.

A szótár típusú kódolások 3 változatát különböztetjük meg:

1. Statikus szótár. Gyakran előforduló frázsokat, digramokat (kétfelüks kombinációkat), vagy  $n$ -gramokat tartalmaz fixen. Nagy mérete lehet és kicsi kompressziója.
2. Félig adaptív szótár. A kódoló létrehoz egy, a tömörítendő üzenethez illesztett szótárat, amelyet együtt tárolunk, illetve együtt továbbítunk az adattal. Ez a technika kétfázisú: az első fázis a szótárat építi, a második fázis pedig az adatot tömöríti. Problema az "optimális szótár" felépítése, amely NP-teljes feladat. Közel optimális algoritmusok azonban léteznek.
3. Adaptív szótár. A szótárat egy menetben felépítik, mialatt az adatokat is kódolják. Nem szükséges a szótárat explicite tárolni vagy továbbítani, mert a dekódoló ugyanúgy felépítheti a szótárat mint a kódoló.

A Lempel-Ziv tömörítők adaptív szótárt használnak. Fontos tulajdonságuk még, hogy bár nem igénylik a forrás statisztikát, bizonyos idő alatt úgy adaptálódnak, hogy az átlagos kódszóhossz/forrásjel minimális lesz. Az ilyen algoritmusokat univerzálisnak nevezzük.

Példa:

[people.inf.elte.hu/birtaivett/Akopjan\\_Alex/LZW](http://people.inf.elte.hu/birtaivett/Akopjan_Alex/LZW)

# The compression effect (from: Ida Mengyi Pu, Fundamental Data Compression, 2006, next 4 slides)

For lossless [compression algorithms](#), we measure the compression effect by the amount of shrinkage of the source file in comparison to the size of the compressed version. Following this idea, several approaches can be easily understood by the definitions below:

- **Compression ratio**

This is simply the ratio of the output to the input file size of a [compression algorithm](#), i.e. the compressed file size after the compression to the source file size before the compression.

Compression ratio = size after Compression / size before Compression

- **Compression factor**

This is the reverse of [compression ratio](#).

Compression factor = size before Compression / size after Compression

- **Saving percentage** This shows the shrinkage as a percentage.

Saving percentage =

(size before Compression - size after Compression) / size before Compression %

Note: some books define the compression ratio as the compression factor defined here. The following example shows how the above measures can be used.

# Veszteséges tömörítések

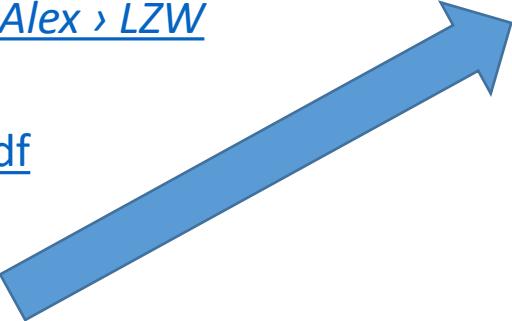
- Válaszoljon a következő kérdésre:
- Hol alkalmazhatóak?
- **Nagy vonalakban** írja le, milyen alapon működnek a:
  - MPEG
  - MP3
  - JPEG

eljárások! (keressen forrásokat a világhálón!)

- Milyen a kompressziós hánnyadosuk (hatékonyságuk)?

# Példák, források:

- Galántai tanár úr jegyzete, 4. fejezet.
- Példa
- [people.inf.elte.hu › birtaivett › Akopjan\\_Alex › LZW](http://people.inf.elte.hu/birtaivett/Akopjan_Alex/LZW)
- Példa és más példák (Huffman)
- <https://ms.sapientia.ro/~kasa/adat13.pdf>
- Történeti áttekintés és példa
- <https://users.iit.uni-miskolc.hu/~lippai/>
- **LZW String tömörítés, kód:**
- <https://juzraai.github.io/blog/2011/lzw-string-tomorites/>
- 



ByteOlvas(Sztring)  
Ciklus amíg nem filevége  
ByteOlvas(Karakter)  
Ha (Sztring+Karakter) benne van a táblában akkor  
Sztring:=Sztring+Karakter  
különben  
Kiír(Sztring)  
TáblaHozzáad(Sztring+Karakter)  
Sztring:=Karakter  
Elágazás vége  
Ciklus vége  
Kiír(Sztring)

# LZW

## Kódolás, szótárkészítés, algoritmus:

1. Beolvassuk az első karaktert (**S**)
2. Beolvassuk a következő karaktert (**X**)
3. Ha **X** a szöveg vége jel, akkor kiírjuk **S** kódját és kilépünk
4. Különben:
  - Ha **SX** benne van a szótárban, akkor **S:=SX**
  - Különben:
    - Kiírjuk **S** kódját
    - Betesszük a szótárba **SX**-et
    - **S:=X**
    - Ugrunk a 2. pontra

## Dekódolás (kitömörítés), algoritmus:

1. Dekódoljuk és kiírjuk az első kódot (**S**)
2. **S\***-ot beírjuk a szótárba
3. Dekódoljuk a következő kódot (**X**)
4. A \* helyére beírjuk x első karakterét
5. Kiírjuk **X**-et
6. **S:=X**
7. Ugrunk a 2. pontra

<https://juzraai.github.io/blog/2011/lzw-string-tomorites/>

Ellenőrizzük a [people.inf.elte.hu › birtavivett › Akopjan\\_Alex › LZW példán!](http://people.inf.elte.hu/birtavivett/Akopjan_Alex/LZW_példán/)

# Feladat

- Írjon fel egy 40 karakter hosszú, legalább 3 különböző jelet tartalmazó szöveget, és kódolja:
  - Huffman kóddal
  - futamhossz kódolással
  - valamely LZ eljárással, leírva a kódolás menetét (megadva természetesen a kódoláshoz használt szótárt is)!
- Hasonlítsa össze az eredményeket!

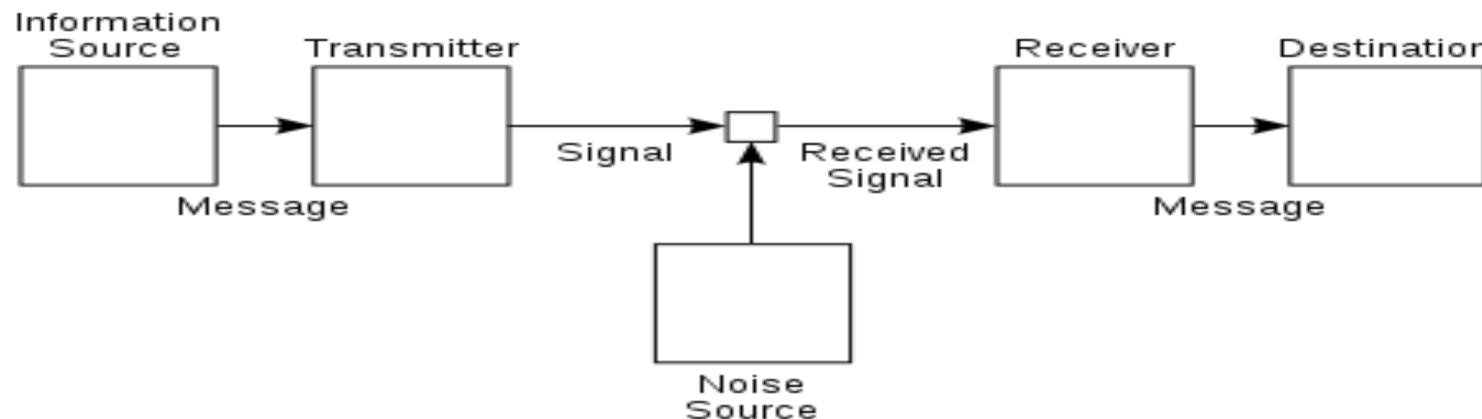
# Titkosítás

Alapfogalmak ismétlése – rövid összefoglaló

9. óra

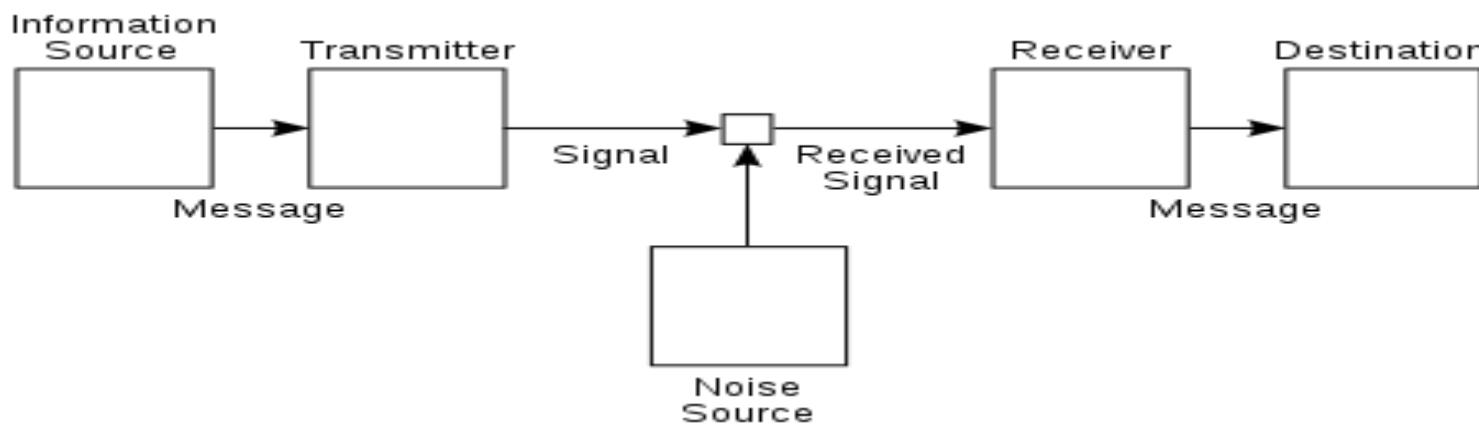
# Mit és hol kódoltunk?

- Kommunikációs csatorna
- Eredeti üzenet kódolása – forráskódolás (Információ és entrópia mérőszámok)
- Csatornakódolás (kölcsönös entrópia, redundáns kódok a továbbítás biztonságáért)



# A csatornakódolás problémái

- Zaj, azaz a csatorna be- és kimenete különbözik (Hamming távolság, hibafelismerő és hibajavító kódok)
- Továbbítás/tárolás költségeinek csökkentése (tömörítési eljárások)
- Titkosítás (adatvédelem az illetéktelen „kódolvasók” aktív vagy passzív támadásaitól a továbbításkor)



„Igazi” biztonságot csak az a titkosítás nyújt, amely tetszőleges mennyiségű választott nyílt szöveg esetén is feltörhetetlen valós időben – gyakorlatban és/vagy elméletben feltörhetetlen titkosítás.

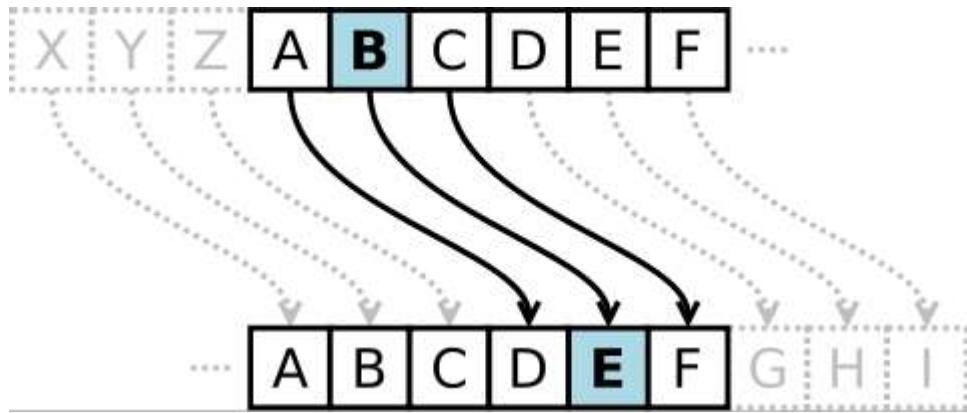
# Általános rövid összefoglaló:

- Tekintettel arra, hogy más tárgyból is tanulják, csak egy rövid fogalom lista – ismerni kell őket nagy vonalakban a második ZH-hoz.
- Az áttekintéshez javaslok:
- [http://www.agr.unideb.hu/~agocs/informatics/01\\_h\\_history/h\\_jpte\\_internet\\_web/amirisc.ttk.pte.hu/network/AJ0901.htm#Titkos%C3%ADt%C3%A1si%20modell%20%C3%A9s%20alapfogalmak](http://www.agr.unideb.hu/~agocs/informatics/01_h_history/h_jpte_internet_web/amirisc.ttk.pte.hu/network/AJ0901.htm#Titkos%C3%ADt%C3%A1si%20modell%20%C3%A9s%20alapfogalmak)

# Fogalomtár

- az elküldeni kívánt eredeti szöveg a nyílt szöveg (plaintext)
- a titkosítási eljárás létrehozza a titkosított szöveget (ciphertext – kriptogram)
- Alapvető eljárások
  - „egyszerű” helyettesítő eljárások
  - kulcs alapú eljárások, kulccsal paraméterezünk  
(A titkosítási kulcs és a megfejtési kulcs nem feltétlenül azonos, de ebben az esetben összetartoznak.)
  - hibrid eljárások (pl. kulcs alapú + keverő algoritmus)
- A megfejtési eljárás eredményeképpen az eredeti nyílt szöveg áll elő.

# Helyettesítő eljárások – Ceasar titkosítás



plaintext: aábcdeéfghiíjklmnoöpqrstuüvwxyz  
ciphertext: ddeéfghiíjklmnoöpqrstuüvwxyzzaábc  
Az írásjelek és a szóközök változatlanul mennek át.

- Általános: k-eltolás
- Általánosítás: a kulcs olyan hosszú, mint az abc, azaz pl. a 26 betűs angol abc betűit helyettesíthetjük az abc betűinek bármely permutációjával
- A lehetséges kulcsok száma:  $26! \approx 4 * 10^{26}$
- Megfejtési idő:  $10^{13}$  év (?)

# Vigenére-féle rejtjelezés

- Az egymást követő jelekből álló nyílt szöveg alá azonos abc-ből származó (ismert) kulcsot alkalmazzunk
- Előnye: könnyen megjegyezhető kulcssql.
- Hátrány: a kulcs hosszának megsejtésekor a probléma visszavezethető az egyábécés titkosításra
- Elegendő titkos szöveg rendelkezésre állásakor statisztikai módszerrel ismét elemezhető - számítógéppel nem probléma
- Áthidalható ez a probléma, ha a kulcs hosszabb, mint a szöveg, de ennek a gyakorlati alkalmazása erősen korlátozott - le kell írni, mert nem megjegyezhető.
- Betűk helyett nagyobb egységek kódolása is lehetséges, például betűpároké vagy szavaké.

# Vigenére-féle rejtjelezés

E R E D E T I S Z O V E G  
+ + + + + + + + + +  
K U L C S K U L C S ...

---

O L....

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

- <https://www.youtube.com/watch?v=SkJcmCaHqS0>
- <https://www.dcode.fr/vigenere-cipher>

# Számítógépes titkosítási módszerek

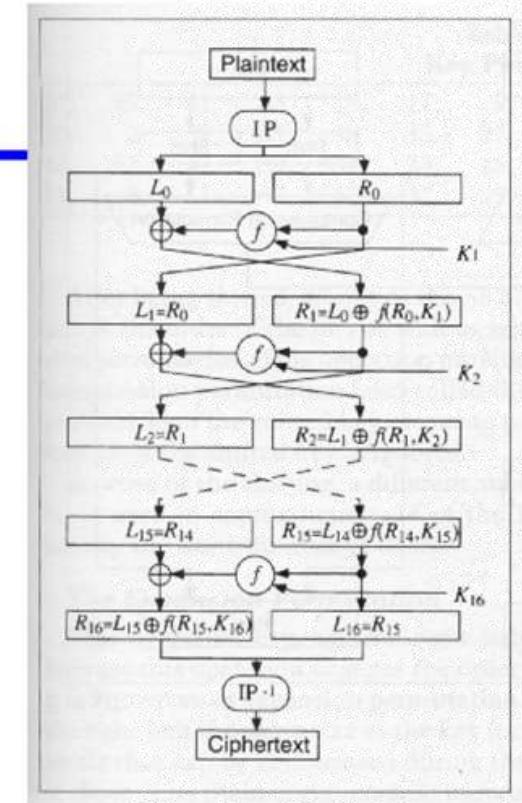
- Mi a cél?
- Akadémiai és üzleti/hadi/... megközelítés
- A széles körben elterjedt platformok és szoftverek feltörése minden napos probléma

# Hibrid módszerek - DES

- USA kormányának 1977-ben elfogadott szabványa a DES.
- 56 bites kulcs parametrizálja, 64 bites blokkokat titkosít 64 bites blokkokká.
- Egyábécés rejtjelezés, amely 64 bites jeleket használ.

## The DES Algorithm

- ◆ 16 identical rounds
  - ◊ Substitution and permutation
  - ◊ Each with a permuted key
- ◆ Details
  - ◊  $IP$  is initial permutation
  - ◊  $L$  is left-half of message
  - ◊  $R$  is right half of message
  - ◊  $\oplus$  is XOR
  - ◊  $K_i$  are keys
  - ◊  $f$  is a function (next slide)
  - ◊  $IP^{-1}$  is an inverse permutation



# Hibrid módszerek - DES

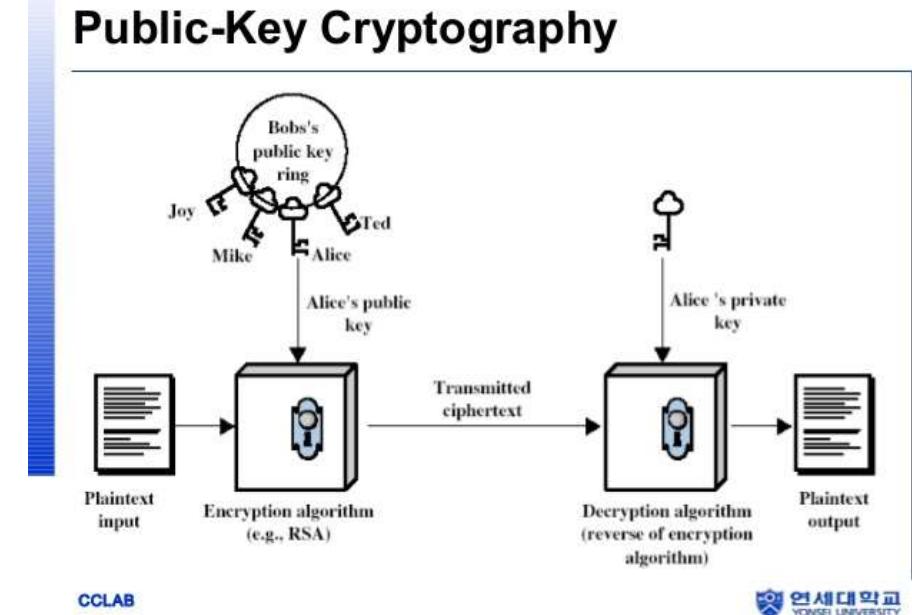
- Az 56 bites kulcs túl rövid, könnyű feltörni (hosszabb kulcs? )
- 3DES - kettő vagy három titkosító kulcsot használ a hagyományos DES által használt egy kulccsal szemben - háromszoros titkosítási eljárást jelent az ilyen eljárást többszörös titkosításnak nevezük.
- „A többszörös titkosítás, több különböző eljárással is megvalósítható. Legegyszerűbb módszere a titkosítási eljárások egymásba skatulyázása. A szöveget titkosítjuk a DES algoritmussal, a kapott első titkosított szöveget egy másik kulcsot használva ismét titkosítjuk, majd a második titkosított szöveget ismét titkosítjuk a harmadik kulccsal. Ez lényegesen megnöveli a titkosítás feltöréséhez szükséges erőforrásigényt, ezzel növeli annak biztonságosságát.
- A 3DES erősebb titkosítást eredményez és ezért a mai napig népszerű és széles körben elterjedt.”\*
- \*[https://hu.wikipedia.org/wiki/Szimmetrikus\\_kulcs%C3%BA\\_rejtjelez%C3%A9s](https://hu.wikipedia.org/wiki/Szimmetrikus_kulcs%C3%BA_rejtjelez%C3%A9s)

# Mi legyen a kulcsokkal?

- A **szimmetrikus kulcsú rejtjelezés vagy titkosítás** lényege, hogy mind a küldő mind a fogadó **ugyanazzal a kulccsal** végzi a titkosítást és a megfejtést.
- minden szimmetrikus kulcsú titkosítás alapja a megosztott titok elve. (példa: DES, 3DES, IDEA)
- A használt titkosítási kulcs különleges eljárást, biztonságot igényel. Ezt speciális kulcskezelési rendszerek támogatják. Problémát okoz, hogy a kulcs egyértelműen feloldja a védett információt.
- Speciális feladat a titkos kulcs küldő és fogadó fél közötti cseréjének a megoldása, a titkosító kulcsok biztonságának védelme a helyi számítógépeken, ... (egyik megoldás a „darabolás”)
- Gyorsak és nincs különösebb erőforrás igényük. Hátrányuk pedig, hogy földrajzilag távol eső helyekre kell teljesen azonos titkosítási kulcsokat eljuttatni és megfelelő biztonsággal tárolni.

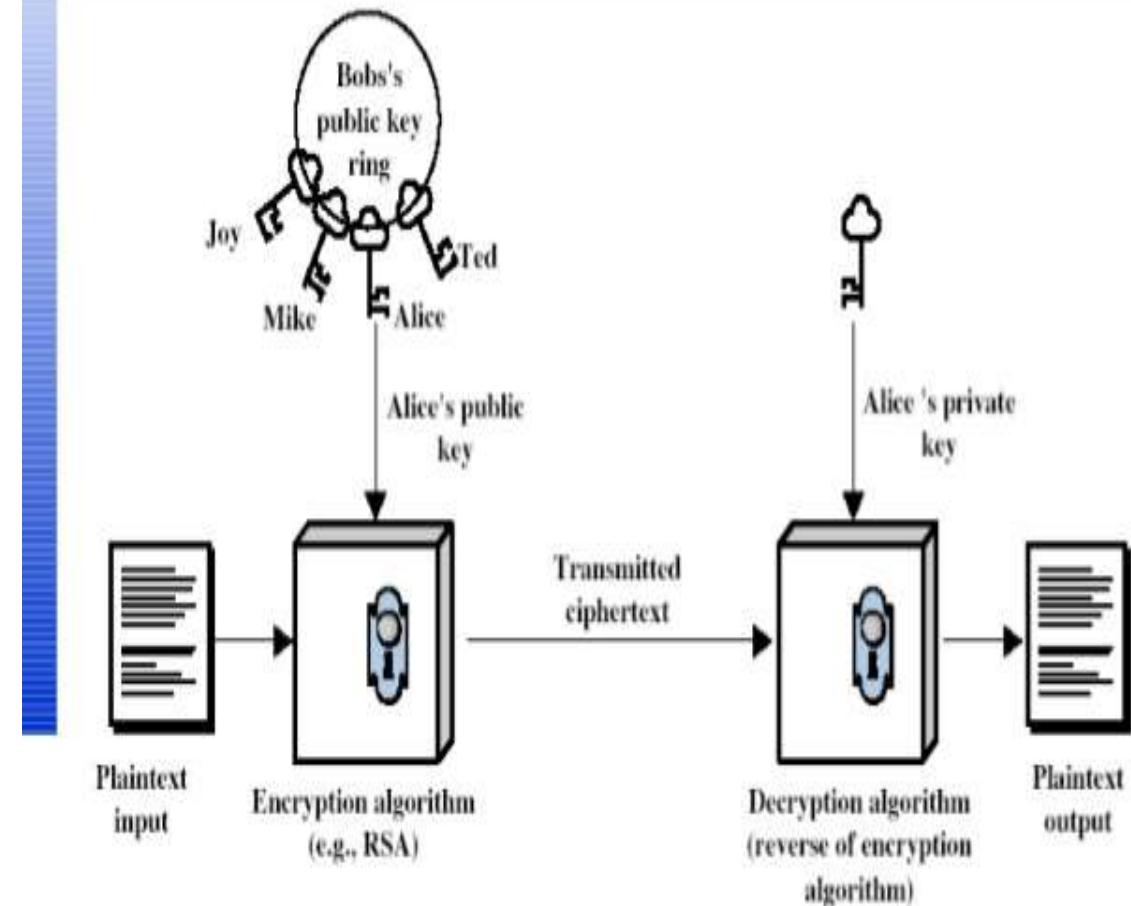
# Nyilvános kulcsok módszere

- RSA eljárás és módszere  
Javasolt forrás pl. :  
<http://slideplayer.hu/slide/2107814/>
- Digitális aláírás
- ...
- Alapvető forrás: Galántai tanár úr jegyzete, 5. fejezet.



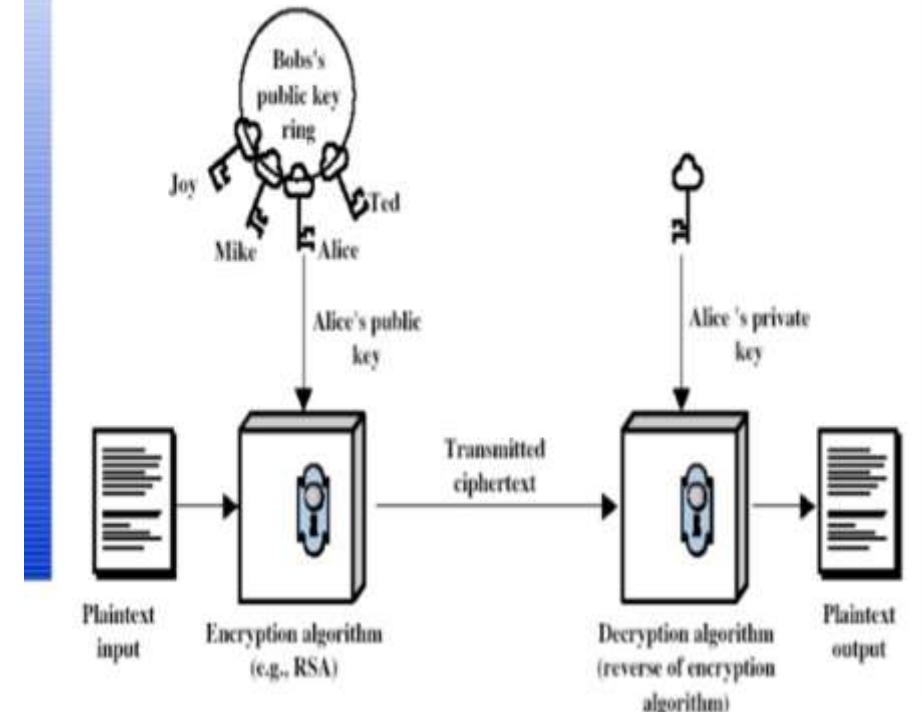
- **Nyilvános kulcsú titkosításnál** minden egyes felhasználóhoz két kulcs tartozik: *egy titkos és egy nyilvános*. A titkos és a nyilvános kulcs szerepe szimmetrikus. Ha N jelöli a nyilvános kulcs alkalmazását, T a titkos kulcsét, és x egy kódolandó információ, akkor
- $N(T(x))=x$  és  $T(N(x))=x$
- A módszer lényege, hogy rendkívül nehéz T-ből N-et meghatározni, továbbá T nem törhető fel választott nyílt szöveggel. Ezeket a követelményeket teljesíti például az MIT-n kidolgozott RSA algoritmus.

## Public-Key Cryptography



- minden felhasználónak generálnia kell a maga részére egy nyilvános/titkos kulcs párt. Ezután a nyilvános kulcsot minél szélesebb körben ismertté kell tenni, a titkos kulcsra értelemszerűen vigyázni kell.
- Bárki, aki titkosított üzenetet akar küldeni, nem kell mászt tegyen, mint a fogadó **nyilvános kulcsával** kódolnia kell az üzenetet. A nyilvános kulcs ismerete nem segít abban, hogy a titkos kulcsot megfejtsük, ezért ha egy üzenetet valaki nyilvános kulcsával kódoltunk, akkor már magunk sem tudjuk azt visszafejteni, csak a fogadó.

## Public-Key Cryptography



CCLAB

# Visszafelé? „Aláírás”

- Ha **hitelesíteni akarunk egy üzenetet**, akkor a saját titkos kulcsunkat használjuk. Az üzenetből képezünk egy az üzenetnél jóval rövidebb számot/üzenetet, amit az üzenet ellenőrző összegének, "ujjlenyomatának" is nevezhetünk. Ezt a számot kódoljuk azután a saját titkos kulcsunkkal. A fogadó ezt csak a mi nyilvános kulcsunkkal tudja "kinyitni" és így biztos lehet abban, hogy az üzenetet valóban mi küldtük.
- Az üzenet ilyen esetben nincs feltétlenül kódolva, de mivel az egész üzenet ujjlenyomatát tartalmazza az aláírásunk, az üzeneten végrehajtott minden változtatás kiderül a fogadó oldalon.
- Ilyen módon - hasonlóan ahhoz, mint amikor aláírunk valamit - a hitelesítéssel nem csak azt garantálhatjuk, hogy kitől származik az üzenet, hanem azt is, hogy az pontosan ugyan az. Ez alkalomadtán - akár csak az aláírás - arra is alkalmas, hogy valamiről bebizonyosodjon, hogy mi hoztuk létre.
- Következésképpen a titkos kulcsunkra nem csak azért kell vigyáznunk, hogy a nekünk küldött üzeneteket ne fejtsék meg illetéktelenek, hanem azért is, hogy mások ne tudjanak „okirathamisítást” végrehajtani a kárunkra.

# RSA algoritmus

- A szerzők nevének kezdőbetűiből: Rivest-Shamir-Adleman “public key criptography”
- Biztonság alapelve: nagy számok tényezőkre bontásának nehézsége. Példa: 200 jegyű szám felbontása a mai számítógépekkel 4 milliárd évig tart.

<http://vassanyi.ginf.hu/info/rsa/index.html>

Válasszunk két nagy prímszámot, (amelyek nagyobbak mint  $10^{100}$  ), majd számítsuk ki a  $n = p * q$  és  $z = (p-1) * (q-1)$  számokat

Legyen  $d$  egy a  $z$ -hez képest relatív prím keressünk egy olyen  $e$ -t, amelyre  $e * d \text{ mod } z = 1$

A titkos kulcs a  $(d, n)$  pár, a nyilvános kulcs az  $(e, n)$  pár

Kódolás  $C = P^e \text{ mod } n$  Dekódolás:  $P = C^d \text{ mod } n$

A kódolás fix méretű blokkokra tördelve történik, a kódolandó blokkok  $\log_2 n$ -nél kevesebb bites egységek.

A kód feltöréséhez  $n$ -t fel kellene bontani  $p$ -re és  $q$ -ra hogy  $z$  és ebből  $d$  meghatározható legyen.

- És még sokan mások...

Válaszolja meg a következő kérdéseket,  
ismételje át a következő fogalmakat!

- Miért fontos a titkosítás?
- Milyen típusú támadások léteznek?
- Történelmi áttekintés:

Átrendezés, behelyettesítés, *Caesar módszer*

Általános behelyettesítés – kulcs bevezetése

Miért probléma a nyelvi statisztika? Miért probléma az egyszeri kulcsos titkosítás?

Vigenère-rejtjel

Enigma – keverő algoritmus résszel bővült

DES, RSA algoritmus főbb jellemzői, lépései, elemei...